



MAX IV/MAX 32 Programmer's Reference Manual

Direct Access Maintenance Processor

211-804014-G00

(

(

(

(

(

(

(

Direct Access Maintenance Processor

SOFTWARE

(

)

C

C

C

C

C

MANUAL HISTORY

Manual Order Number: 211-804014-G00

Title: MAX IV/MAX 32 PROGRAMMER'S REFERENCE MANUAL, DIRECT ACCESS
MAINTENANCE PROCESSOR

Software Part Numbers: 611599-113G.0

Revision Level	Date Issued	Description
--	07/76	Initial Issue.
--	04/77	Revision.
--	08/77	Revision.
--	12/77	Reissue.
--	02/78	Revision.
--	05/78	Revision.
--	06/79	Reissue (F.1).
--	11/79	Reissue (F.2).
F01	02/82	Reissue (F.3). Manual order number changed from 210-600500-015.
F02	12/82	Reissue (F.4).
G00	07/85	Reissue (G.0). Product number changed, MAX III references deleted, MAX 32 added. Formatted to current standards.

Contents subject to change without notice.

Copyright © 1976, by Modular Computer Systems, Inc.
All Rights Reserved.
Printed in the United States of America.

PROPRIETARY INFORMATION

THIS MANUAL CONTAINS CONFIDENTIAL PROPRIETARY INFORMATION PROTECTED BY SOFTWARE PROPERTY RIGHTS AND IS MADE AVAILABLE UPON THE CONDITION THAT THE SOFTWARE CONTAINED HEREIN WILL BE HELD IN ABSOLUTE CONFIDENCE AND MAY NOT BE DISCLOSED IN WHOLE OR IN PART TO OTHERS WITHOUT THE PRIOR WRITTEN PERMISSION OF MODULAR COMPUTER SYSTEMS, INC.

FOR GOVERNMENT USE THE FOLLOWING SHALL APPLY

RESTRICTED RIGHTS LEGEND

USE, DUPLICATION OR DISCLOSURE BY THE GOVERNMENT IS SUBJECT TO RESTRICTIONS AS SET FORTH IN PARAGRAPH (b) (3) (B) OF THE RIGHTS IN TECHNICAL DATA AND COMPUTER SOFTWARE CLAUSE IN DAR7-104.9(a) (OR SUCH EQUIVALENT FPR, NASPR, AND/OR DOEPR CLAUSE AS MAY BE APPLICABLE).

MODULAR COMPUTER SYSTEMS, INC.
1850 WEST McNAB ROAD
FORT LAUDERDALE, FL 33309

PREFACE

Audience

This manual is directed to users of FORTRAN direct access facilities.

Subject

This manual describes the creation and maintenance of direct access sub-partitions for use by FORTRAN code.

Product Requirements

This processor runs under the MAX IV and MAX 32 operating systems.

Related Publications

Refer to the following manuals for additional information. When ordering manuals, use the manual order number listed below. The most current revision level (REV) will be shipped.

<u>Manual Order Number</u>	<u>Manual Title</u>
213-804001-REV	MAX IV GENERAL OPERATING SYSTEM System Guide (SGM)
213-804001-REV	MAX 32 GENERAL OPERATING SYSTEM System Guide (SGM)
210-804002-REV	MAX IV FORTRAN IV Language Reference Manual (LRM)
211-804002-REV	MAX IV/MAX 32 NONRESIDENT JOB CONTROL Programmer's Reference Manual (PRM)
211-804008-REV	MAX IV/MAX 32 LIBRARY UPDATE Programmer's Reference Manual (PRM)

MODCOMP Training Department

MODCOMP's Education Services department provides training courses on many hardware and software products at our training Center in Fort Lauderdale, Florida. On-site courses at customer sites can also be arranged. For more information about the courses offered by Education Services, contact MODCOMP Training Registrar.

(

4

(

(

(

1

TABLE OF CONTENTS

	PAGE
CHAPTER 1 OVERVIEW OF THE DIRECT ACCESS MAINTENANCE PROCESSOR	
1.1 GENERAL DESCRIPTION.....	1-1
1.2 FUNCTIONAL DESCRIPTION.....	1-1
 CHAPTER 2 SUMMARY OF DIRECTIVES	
2.1 JOB CONTROL DIRECTIVE LIST	2-1
2.2 DIRECTORY MANIPULATION DIRECTIVES	2-1
2.3 DIRECTIVES REFERENCING DATA SETS	2-2
2.4 OPERATIONAL CONTROL DIRECTIVES	2-2
 CHAPTER 3 DIRECTIVES	
3.1 GENERAL INFORMATION AND SYNTAX	3-1
3.2 JOB CONTROL DIRECTIVE LIST	3-2
3.3 ALL OTHER DIRECTIVES	3-3
ALLOCATE	3-3
APPEND	3-5
ASCII	3-6
COMPRESS	3-7
COPY	3-8
CREATE	3-9
DEFINE	3-10
DISPLAY	3-11
ELIMINATE	3-13
ENTER	3-14
EXIT	3-15
INITIALIZE	3-16
LIST	3-17
MIMIC	3-18
MOVE	3-19
NOVERIFICATION	3-20
ONE	3-21
PURGE	3-22
REFILE	3-23
REMOVE	3-24
RENAME	3-25
RESTORE	3-26
SAVE	3-28
SET	3-30
TWIN	3-32
TWO	3-33

CHAPTER 4 GUIDE TO THE USE OF DAMP

4.1 FILE USAGE	4-1
4.2 KEYS	4-1
4.3 REFERENCING DATA IN A DATA SET	4-2
4.4 CHAINED RAD PARTITIONS	4-3
4.4.1 General Information	4-3
4.4.2 FORTRAN Mechanism	4-4
4.4.3 Usage	4-7
4.4.4 Use of RERAD	4-7
4.5 SYSTEM CONFIGURATIONS	4-7
4.6 SPECIAL DATA FORMATS	4-8
4.7 SAVE/RESTORE ARCHIVAL FORMAT	4-9
4.8 LIMITATIONS	4-10
APPENDIX A ERROR MESSAGES	A-1
APPENDIX B EXAMPLES FOR USING DAMP	B-1

FIGURES

3.1 FORTRAN RUN-TIME DATASET LOCATION	4-6
---	-----

CHAPTER 1 OVERVIEW OF THE DIRECT ACCESS MAINTENANCE PROCESSOR

1.1 GENERAL DESCRIPTION

The Direct Access Maintenance Processor (DAMP) is used to perform utility operations on arbitrary sequential files, manipulate FORTRAN direct access directories recorded on file RAD, and to perform utility direct access file operations whose results can be modified by information from the direct access directory. DAMP is primarily used for direct access file processing.

DAMP allows you to do the following:

- . Create a new direct access directory.
- . Change information in, remove information from, or add information to a directory.
- . Concatenate discrete directories into a single directory.
- . List the contents of the current directory.
- . Move directories from file to file.
- . Display selected records from a file.
- . Copy files to other files.
- . Place duplicate values in each word of selected records of a file.
- . Save files and restore them later.
- . Establish a chain of Random Access Directories for use by FORTRAN programs.

1.2 FUNCTIONAL DESCRIPTION

The user specifies the operations to be performed by DAMP with a sequence of directives which the program reads from the CI (command input) file. The specified operations are performed one at a time in the order the directives are read.

There are basically four areas in which DAMP directives can be categorized, Job Control directives that can be used in DAMP as well as Job Control, directory manipulation directives, directives that reference data sets, and operational directives that are used to control the operation of the DAMP program. The format of each directive and a complete definition of the function of each directive is listed in Chapter 3.

(

(

(

(

(

(

(

CHAPTER 2 SUMMARY OF DIRECTIVES

The directives recognized by DAMP fall roughly into four categories: directives which duplicate MAX IV/MAX 32 Job Control functions, directives which manipulate the user's data set directory on the RAD file, directives which reference data sets, and directives which control certain internal functions of DAMP. A brief discussion of these four categories follows. For detailed information on directives, refer to Chapter 3.

2.1 JOB CONTROL DIRECTIVE LIST

The following directives recognized by Job Control can also be used under DAMP.

\$NOP	ASSIGN
ACTION	AVRECORD
AVFILE	AVFILE
BKFILE	WEOF
REWIND	

These directives are described in Chapter 3. Refer to the MAX IV/MAX 32 NONRESIDENT JOB CONTROL Programmers Reference Manual for detailed information.

2.2 DIRECTORY MANIPULATION DIRECTIVES

These directives are used primarily to manipulate your data set directory recorded on the file to which the name RAD is currently assigned. All of these directives except CREATE and MOVE require that a valid directory exist on the RAD file. MOVE requires that a valid directory exist on the file named after the word MOVE.

ALLOCATE	searches the directory entries of a specified file to find currently unused space, calculates space needed by data set being allocated, creates a directory entry, and can assign to a new data set space overlapping the space assigned to another set.
APPEND	copies the contents of one directory after the end of another directory.
CREATE	writes a new, empty directory on the RAD file.
ELIMINATE	allows selective deletion of any empty directories.
ENTER	places a specific entry into the data set directory.
INITIALIZE	uses the size of the directory already existing to determine the number of empty entries to generate.
LIST	lists the entire contents of the current directory.
MOVE	copies a directory to another file.
PURGE	deletes all empty entries from the directory.

REFILE	changes the filename portion of every directory entry which references that file.
REMOVE	deletes the entry for a data set from the directory.
RENAME	changes a specified data set key in a directory entry to some other key. It can also be used to change the file name in that particular directory entry.
TWIN	places a new entry in the user's directory on RAD which duplicates the original entry.

2.3 DIRECTIVES REFERENCING DATA SETS

The following directives are used primarily to manipulate data sets or individual records in data sets. All of these directives are profoundly influenced by the directory contents and COMPRESS and RESTORE usually modify the directory.

COMPRESS	rewrites an entire file and releases all unused space by collecting it into a contiguous area at the beginning of the file's partition.
COPY	serves to copy data sets from one file to another.
DISPLAY	provides a listing of individual records from a data set.
RESTORE	regenerates the data saved.
SAVE	causes the user's directory on file RAD ,the current core DEFINE table, and/or one or more data sets to be written in a special format suitable for archival storage.
SET	places a specified value into every word of a record or a sequence of records in a data set.

2.4 OPERATIONAL CONTROL DIRECTIVES

The following directives are used to control the operation of the DAMP program and the way in which it interprets other directives.

ASCII	allows the COPY directive to read ASCII files instead of binary files.
DEFINE	creates a table (DEFINE table) and stores the necessary parameters into the table for use by other directives.
EXIT	causes DAMP to relinquish control to the executive. It should be entered when the current operations under DAMP have been completed.
ONE	outputs 80 characters per line when using the DISPLAY directive.
TWO	outputs 40 characters per line when using the DISPLAY directive. Used if LO is assigned to a console device.

CHAPTER 3 DIRECTIVES

3.1 GENERAL INFORMATION AND SYNTAX

DAMP uses several directives, some of which are also Job Control directives. This manual assumes familiarity with these Job Control directives and presents only a brief description of their use and syntax for reference. If more information is needed, refer to the MAX IV/MAX 32 Nonresident Job Control Programmer's Reference Manual. All other directives are in alphabetical order and include a functional description, proper syntax, and examples of use. The keyword in all directives can be abbreviated to the first three characters.

Many of the directives require file names and/or data set keys as parameters. File names can consist of from one to three MODCOMP CAN characters. Leading and embedded blanks are not permitted. The key is the actual 16-bit quantity used to identify a specific data set. Most of you will specify this key as a numeric quantity, that is, as a data set reference number. Data set keys can be entered in one of three formats. These conventions for naming data sets are discussed in section 4.2. Neither file names nor data set keys can have a binary value of zero.

NOTE: The following directives require that a valid data set directory be present on the initial sectors of the partition to which the name RAD is currently assigned before they can be entered:

ALLOCATE	APPEND	COMPRESS
ELIMINATE	ENTER	INITIALIZE
LIST	PURGE	REFILE
REMOVE	RENAME	TWIN

The general syntax for all directives is:

```
function  [logfilename][,options]
          [filename]
          [text]
          [key]
          [integer]
```

function Parameter 1 (required) specifies the directive name of the function. Only the first three characters of the directive name are significant.

 Parameter 2 (optional) specifies one of the following:

logfilename logfilename specifies a logical file name.

filename filename specifies any valid filename.

text text specifies a message directed to the operator's attention.

key key specifies any valid data set name.

integer integer specifies non-negative numbers.

3.3 ALL OTHER DIRECTIVES

ALLOCATE

Allocates Space on a Specific Filename and Adds an Entry to the Directory on the RAD File.

The ALLOCATE directive searches the directory entries of a specified file to find currently unused space.

This directive will allocate space on the specified "filename" large enough to contain a data set with the parameters given and will add a suitable entry to the directory on the user's RAD file. The data set parameters are also entered into the DEFINE table (refer to the DEFINE directive).

This directive is useful when the parameters of the FORTRAN DEFINE FILE statement for a data set are known and when the set can be placed in the next unused portion of the file being sub-partitioned.

SYNTAX

ALL[OCATE] filename,key-1,integer-1,integer-2 [,format-indicator][,key-2]

filename	Parameter 1 (required) specifies the MAX IV or MAX 32 name of the file in which the new data set will reside. This name must be currently assigned to a disc partition. The name RAD can be used if the data set is to share space with the user's directory.
key-1	Parameter 2 (required) specifies the data set name to be used when referencing this subfile.
integer-1	Parameter 3 (required) specifies the number of records in this data set.
integer-2	Parameter 4 (required) specifies the maximum size of each record.
format-indicator	Parameter 5 (optional) specifies any parameter that will be identified as such by the executive's text string scanning services. If the parameter is specified and starts with the letter "U", the record size, "integer-2," is specified in units of MODCOMP words. If "format-indicator" is missing or does not start with "U", the record size is assumed to be in bytes.
key-2	Parameter 6 (optional) specifies the data set whose space is to be shared with the new data set. If it is present, the new data set will start with the first sector of the specified data set. If it is missing, the new data set will occupy a previously unused portion of "filename."

EXAMPLE

ALLOCATE MFL,BH,108,25

The RAD directory is searched to find the lowest sector in use on file MFL. Enough sectors are reserved before this sector to contain 108 thirteen-word records. (Fifteen 100-word sectors or eleven 128-word sectors will be reserved.) An entry describing the new data set is added to the RAD directory. The key for this entry is 16,968 (hexadecimal 4248 or the CAN-code of JXH). An entry is added to the DEFINE table specifying that data set BH contains 108 thirteen-word records.

Copies the Contents of One Directory After the End of Another Directory

The APPEND directive is used to copy the contents of one directory after the end of another directory.

The data set directory recorded in the initial sectors of "filename" is appended to the directory on RAD. A sector which contains the name "filename" but which does not contain directory entries is added to RAD to delimit the original directory. Refer to the ELIMINATE directive. Unused entries and directory sectors which do not contain any entries will not be transferred. Both RAD and "filename" must contain valid directories before this directive is issued. If "filename" is assigned to a sequential device, that device must be rewindable and the directory must be written at the rewind position of the device medium. If the directory on "filename" contains references to sub-files assigned to RAD on a partition different from that of the current RAD, then file name RAD should be changed before appending "filename." Refer to the REFILE directive.

SYNTAX

APP[END] filename

filename Parameter 1 (required) specifies any valid file name except RAD. If RAD is specified, an error occurs.

EXAMPLE

APPEND MFL

The directories on RAD and on MFL are searched for references to RAD, and the sum of the directories' lengths is checked to be sure it is less than the lowest RAD sector used. Then the final continuation bit in the RAD directory is set to 1 and an "empty" sector containing the CAN-code for MFL is written after the RAD directory. Finally, the directory on MFL is reblocked and written on the next sectors of RAD. Empty entries and sectors are not transferred.

ASCII

Indicates to the Next Copy Directive that ASCII Format Is Used to Read Data.

The ASCII directive indicates to the next COPY directive that ASCII format is to be used for the reading of data.

SYNTAX

ASC[II]

Packs Data Sets Into Contiguous Disc Sectors

The COMPRESS directive can be used on a file whenever there is not enough space left in the file to accommodate new data sets and when some data sets previously residing on the partition have been deleted. It recopies all data sets currently listed for the specified file to contiguous sectors at the end of the file's partition.

All data sets contained in "filename" are packed into contiguous disc sectors and the directory on RAD is updated. This will allow the ALLOCATE directive to use all unused portions of "filename" for new data sets. Both "filename" and RAD must be assigned to appropriate disc partitions before issuing the COMPRESS directive.

The directory on RAD is searched for all entries describing data sets residing on file SC. If there are any unused portions of SC with sector addresses higher than the addresses of some used portions, the data in the lower used portions will be rewritten in the higher sectors and their directory entries will be modified. This is used on a file whenever there is not enough space left in the file to accommodate new data sets and some data sets previously residing on the partition have been deleted.

SYNTAX

COM[PRESS] logfilename

logfilename Parameter 1 (required) specifies the name of the logical file containing the data sets to be packed. References to "filename" should appear in the current user's directory on RAD.

COPY

Copies a Data Set From One File to Another

The COPY directive serves to copy data sets from one file to another. Copy can operate on sequential disc or magnetic tape files as well as on direct access data sets. Since it is primarily intended for data sets, it always obtains record size information from you or the system device tables. It never uses the byte count returned to it from the Executive READ Service. Thus, sequential files are treated as if they contained fixed length records, true record lengths are ignored.

The data set specified by "key-1" is copied to the data set specified by "key-2." The format of the data transfers and the number of records copied depends on the type of device containing the file which contains the data set, on whether the data set is in the user's directory on RAD, and on whether the data set is in the current DEFINE table. All of the possible cases are discussed in section 4.3. Copying begins with record 1 and continues to the end of the shorter data set. The number of records actually written is listed on LO following the listed COPY directive.

SYNTAX

COP[Y],key-1,key-2

key-1 Parameter 1 (required) specifies the data set key to be copied.

key-2 Parameter 2 (required) specifies the data set key to copy to.

EXAMPLE

COPY BI,SI

The data set with key 16,769 (equivalent keys: BI and @JXI) is copied to the data set with key 21,321 (equivalent keys: SI and @MMA). Copying stops when the smaller data set has been completely referenced. The actual number of records written is printed on LO.

Generates a User's Directory of Empty Entries on RAD

The CREATE directive is used to write a new, empty directory on the RAD file. A parameter is used to specify the number of empty entries to be placed in the new directory. In most cases, this should be 1. If you want to reserve a few sectors after the directory, this parameter can be increased appropriately.

This directive generates and records on RAD a user's directory containing "integer" empty entries. RAD **MUST** be assigned to a disc partition before this directive is used. A 100-word sector can accommodate 24 entries and a 128-word sector can accommodate 31. If you designate this quantity accommodated by A, the number of sectors written on RAD by CREATE will be given by the first integer greater than or equal to "integer" divided by the A.

If this directive is not preceded by a NOVERIFICATION directive, the program will go into the HOLD state after writing CREATE RAD DIRECTORY REQUESTED on the CO file. The operator can then depress the CONSOLE INTERRUPT switch and type /RES to continue or else abort the job. If NOVERIFICATION is specified before this directive, the program will output the message preceded by the word UNVERIFIED and continue to the next directive. After each CREATE directive the program will default back to the HOLD state.

NOTE: Excess entries can increase the time needed to search the directory.

SYNTAX

CRE[ATE] integer

integer	Parameter 1 (required) specifies the number of empty entries to be created on a directory. If "integer" is zero, a "dead" empty sector will be placed on RAD. No information will ever be placed into the sector, but it will still be accessed whenever RAD is referenced. The impact of this wasted access can be considerable if the partition does not have a suitable geometry.
---------	--

DEFINE

Creates a Table Containing Parameters and Stores Them For Use By Other Directives

The DEFINE directive places the specified configuration parameters for the data set identified by "key-1" into a table for use by the COPY, DISPLAY, SAVE, and SET directives. The DEFINE table can contain a maximum of ten entries. The latest ten entries with unique keys are retained. The ALLOCATE directive also introduces the characteristics of the new data set into the DEFINE table.

SYNTAX

DEF[INE],key-1,integer-1,integer-2 [,format-indicator]

key-1	Parameter 1 (required) specifies the the data set key with the parameters to be entered into the the table.
integer-1	Parameter 2 (required) specifies the number of records in key-1.
integer-2	Parameter 3 (required) specifies the number of words in each record.
format-indicator	Parameter 4 (optional) specifies any parameter that will be identified as such by the executive's text string scanning services. If the parameter is specified and starts with the letter "U", the record size, "integer-2," is specified in units of MODCOMP words. If "format-indicator" is missing or does not start with "U", the record size is assumed to be in bytes.

EXAMPLE

```
DEFINE SC,25,38,E
```

An entry is made in the DEFINE table associating key 21,315 (equivalent keys: SC and @ML8) with 25 records, each containing 19 words.

Provides a Listing of Individual Records from a Data Set.

The DISPLAY directive provides a listing of individual records from a data set. Both the initial record position and the number of records to be printed can be specified when it is used. It can operate with sequential files but remember that certain restrictions apply. Refer to section 4.3.

This directive prints "integer-2" records from the data set specified by "key," starting with record number "integer-1."

If "integer-1" is missing, the display will begin with record 1. If "integer-2" is missing, the display will continue to the end of the set. In ASCII or hexadecimal format, the amount of information printed by DISPLAY is dependent on the device containing the file containing the data set, on whether the data set is in the DEFINE table, and on whether the data set appears in the user's directory on RAD. These factors are later discussed in section 4.3. Refer also to the directives ONE and TWO. If integer or floating-point format is selected, a maximum of five numbers will be output per line. If the TWO directive has been used two numbers will be output per line.

Both the initial record position and the number of records to be printed can be specified when it is used. The information will be displayed in ASCII or in hexadecimal as specified by a directive parameter. Like COPY, DISPLAY can operate with sequential files. Many of the same restrictions apply to both directives. The way in which record lengths, starting position, and the like are determined for these directives and for SET is discussed in section 4.3.

The display format has various options which can be compiled into DAMP. The format indicator and options are as follows:

<u>INDICATOR</u>	<u>OPTIONS</u>
Always available	
A or null	= ASCII
H	= Hexadecimal
IRD OPTION	
I	= 16-bit integer
R	= Real, 32-bit floating point
D	= Double precision floating point (48-bit)
LE option	
L	= Long integer (32-bit)
E	= Extended floating point (64-bit)

SYNTAX

DIS[PLAY] key [integer-1][,integer-2][,format-indicator]

- | | |
|--------------------|--|
| key | Parameter 1 (required) specifies the data set key that contains the records to be displayed. |
| integer-1 | Parameter 2 (optional) specifies the record number to start displaying from. If not specified, it will start from the first record. |
| integer-2 | Parameter 2 (optional) specifies the number of records to be displayed. If not specified, everything to the end of the set will be displayed. |
| [format-indicator] | Parameter 4 (optional) specifies any parameter that will be identified as such by the executive's text string scanning services. If the parameter is specified and starts with the letter "U", the record size, "integer-2," is specified in units of MODCOMP words. If "format-indicator" is missing or does not start with "U", the record size is assumed to be in bytes. |

EXAMPLES

- | | |
|--------------------|---|
| DISPLAY 20,1,10,A | Display data set 20, records 1 to 10 in ASCII. |
| DISPLAY 30,,,L | Display the entire data set 30 in 32-bit integer format. |
| DISPLAY 40,10,20,R | Display data set 40, starting at record 10 for 20 records in single precision floating point. |

ELIMINATE

Allows Selective Deletion of Any Empty Directories

The ELIMINATE directive allows selective deletion of any empty directories.

The directory on RAD is searched for a sector which has no entries but which has been flagged with the name "filename." This "empty" sector and all subsequent directory sectors, either to the next "empty" sector or through the end of the directory, are deleted from the directory.

SYNTAX

ELI[MINATE] filename

filename Parameter 1 (required) specifies any file name which has previously been used in an APPEND directive.

EXAMPLE

ELIMINATE SCA

The directory on RAD is scanned for "empty" sectors. All such sectors are examined to see if they contain the CAN-code for SCA in word 1. When such a sector is found, it and all subsequent sectors are skipped until another "empty" sector is found. This sector and all subsequent sectors to the end of the directory are rewritten starting at the sector where the "empty" sector flagged with the SCA was found. If a second "empty" sector is not found, the sector before the flagged sector is rewritten with a zero initial bit.

ENTER

Places a Specific Entry into the Data Set Directory

The ENTER directive is used to place a specific entry into the data set directory. No file searching or computation is done. This directive is used when it is desired to specify exactly what an entry is to contain.

The specified information will be inserted into the directory recorded on RAD. This directive does not affect the DEFINE table.

The ENTER @\$\$ form defines a chained RAD entry. When a FORTRAN program is executed, the logical file associated with the entry will be searched as RAD for the required dataset. (Refer to Chained RAD Partitions). ENTER @\$\$x RAD causes a FILE TOO BIG error consistent with the attempt to create an infinite chain. The directive is used when it is desired to specify exactly what an entry is to contain.

SYNTAX

ENT[ER] key,filename,[integer-1][,integer-2]

key	Parameter 1 (required) specifies the data set key.
filename	Parameter 2 (required) specifies the file name of the new entry.
integer-1	Parameter 3 (optional) specifies the starting point of the sector. "Integer-2" should not be less than "integer-1". The default is zero.
integer-2	Parameter 4 (optional) specifies the number of sectors assigned. "Integer-2" should not be less than "integer-1." The default is zero.

or

key	Parameter 1 (required) specifies @\$\$q, where q is any valid CAN-codeable character.
filename	Parameter 2 (required) specifies the logical file name of the new entry.
integer-1	Parameters 3 and 4 (optional) can be omitted and are always forced to zero.

EXAMPLES

ENTER 2,2,0,1023

This directive adds an entry to the directory on RAD. This entry described a data set with a key of 2, which occupies the first 1024 sectors of the partition to which the user assigned the file name 2.

ENTER @\$\$: LNK

The entry is added into the current RAD directory with a start and end sector of zero. The FORTRAN Run Time Library will use this entry to cause the RAD format partition, to which LNK is assigned at the time of the run, to be searched for data sets required. (Refer to Chained RAD Partitions).

EXIT

Terminates Execution

The EXIT directive terminates execution of DAMP. If it was executing in the background, Job Control resumes execution.

SYNTAX

EXI[T]

INITIALIZE

Clears the Directory on RAD

The INITIALIZE directive clears all entries from the user's directory on RAD. Directory sectors which do not contain entries are not modified. This directive is similar to CREATE, but the directory size is not altered. The data sets previously listed in the directory are not physically modified, but they cannot be retained unless appropriate new allocations are made in the directory.

If this directive is not preceded by a NOVERIFICATION directive, the program will go into the HOLD state after writing INITIALIZATION OF RAD DIRECTORY REQUESTED on the CO file. The operator can then depress the CONSOLE INTERRUPT switch and type /RES to continue or else abort the job. If NOVERIFICATION is specified before this directive, the program will output the message preceded by the word UNVERIFIED and continue to the next directive. After each INITIALIZE directive the program will default back to the HOLD state.

SYNTAX

INI[TIALIZE]

Lists the Entire Contents of the Current Directory

The LIST directive prints the user's directory, one entry per line. The data set key is printed three times, first in hexadecimal, next in decimal, and next "uncanned." The file name is listed next, followed by the starting and ending sector numbers, in decimal. The last column is the number of sectors for the file in decimal.

It is useful to verify the results of previous operations. Since keys can have been entered in different ways, the key portion of each directory entry is listed in hexadecimal format, in decimal format, and as the three ASCII characters generated by applying the executive CAN-to-ASCII service to the key.

SYNTAX

LIS[T] DIRECTORY

MIMIC

Searches for a Key and Copies the Entry to the Current RAD File

The MIMIC directive will search the directory in filename-2 for the specified "key" with file name-1. When an entry is found for this "key" the entire entry will be copied to the current RAD directory. If the optional (filename-3) is present, the new entry will have an alias file name as specified in the optional parameter.

The MIMIC command duplicates the entry from RAD filename-2 into the current RAD file. Subsequent changes to filename-2 are not repeated in the current RAD file. Therefore, care is required when using the RAD files which have had entries copied. In particular, the effects of COMPRESS (refer to section 4.8 Limitations) cause serious problems. The MIMIC command warns of these problems, and the solution by the messages

MIMIC: COMPRESS MUST NOT BE USED
ALTERNATIVE: ENTER @\$X LFN (REF. MANUAL)

Use of chained RAD files avoids these difficulties. (Refer to section 4.5 Chained RAD Partitions)

SYNTAX

MIM[IC] key,filename-1,filename-2[,filename-3]

- key Parameter 1 (required) specifies any valid data set keys.
- filename-1 Parameter 2 (required) specifies any valid file names.
- filename-2 Parameter 3 (required) specifies any valid file names.
- filename-3 Parameter 4 (optional) specifies any valid file names.

EXAMPLE

MIMIC 2 SCB SCA DNB

The SCA directory is searched for a data set key of 2 with a file name of SCB. This entire entry will be entered in the currently assigned RAD directory. The optional file name DNB will be used as an alias file name in the RAD directory.

Copies A Directory to Another File

The directory recorded on "filename" is copied onto RAD. Empty entries are not transferred, and reblocking occurs if the record sizes of the two files differ. If "filename" is assigned to a sequential device, the directory must appear as the next accessible records.

NOTE: "Filename" must contain a valid directory before this directive is used.

SYNTAX

MOV[E] filename[,deletion-indicator]

filename	Parameter 1 (required) specifies the file name with the directory to be copied onto RAD.
deletion-indicator	Parameter 2 (optional) specifies any parameter which the executive text scanning services will recognize as such. If the "deletion-indicator" is present, any of the directory records on "filename" which do not contain records will not be transferred to RAD. If the indicator is omitted, these "empty" records will be transferred on a record-to-sector copying basis, with no reblocking.

EXAMPLES

MOVE MTF,NO

The directory on file MTF is recorded on RAD, with suitable reblocking and with all unused entries deleted. In addition, because of the NO, "empty" sectors are dropped and reblocking occurs as if "empty" sectors did not appear.

NOVERIFICATION

Allows Execution of a Directive Without Operator Intervention

This directive should precede the INITIALIZE, or CREATE directives to prevent a HOLD condition after the message is output to the CO file. This allows execution of a directive without operator intervention, whereby the program will automatically proceed to the next directive. This directive will cause the message NOVERIFICATION to be output to the CO file.

SYNTAX

NOV[ERIFICATION]

Prints Output From DISPLAY Directive 80 Characters Per Line

The ONE directive causes the DISPLAY directive to print according to the following table. ONE is in effect by default.

Display format ONE

A	80 characters
H	16 hexadecimal words
I	5 numbers
R	5 numbers
L	5 numbers
D	5 numbers
E	5 numbers

SYNTAX

ONE

PURGE

Deletes All Empty Entries from the RAD Directory

The PURGE directive causes any unused entries in the current RAD directory to be deleted and successive sectors to be consolidated if possible. If sectors which do not contain entries appear in the directory, any partially filled sector built so far will be written, exact copies of the sector(s) which do not contain entries will be written next, and purging will resume with the next input sector containing entries (if any). Thus, a directory containing appended directories (refer to APPEND) can be purged without harm.

NOTE: A null directory will not be created by PURGE.

SYNTAX

PUR[GE]

Changes the File name Portion of Every Directory Entry Which References the File.

The REFILE directive changes the file name portion of every directory entry which references that file. It is especially useful if a single directory is being created from several discrete directories.

The directory on RAD is searched for any data sets residing in "filename-1." If any are found, their file names are changed to "filename-2." Their keys and sector limits are not modified.

SYNTAX

REF[ILE] filename-1,filename-2

filename-1 Parameter 1 (required) specifies the file name to be searched.

filename-2 Parameter 2 (required) specifies the file name to change the data sets to.

REMOVE

Deletes an Entry from the Directory

The REMOVE directive removes the entry for the data set specified by "key" from the user's directory and produces an empty entry.

SYNTAX

REM[OVE] key

key Parameter 1 (required) specifies the data set name to be deleted.

EXAMPLES

```
REMOVE 6
REM 80
REM @271
REMOVE LP
```

The directory entry for the data set with key 19,536 (equivalent keys: LP and @LHP) is replaced on RAD with an entry containing zeros in its first two words.

RENAME

Changes a Specified Data Set Key in a Directory Entry to Some Other Key

The RENAME directive changes a specified data set key in a directory entry to some other key.

The directory on RAD is searched for a data set designated by "key-1" and changes it to key-2. If "filename" is present, the file name associated with "key-1" is also changed. To change only the file assignment, "key-1" and "key-2" should be identical. This directive does not modify sector limits. To change the file name for all subfiles on a given file, use REFILE.

SYNTAX

REN[AME] key-1,key-2,[,filename]

key-1 Parameter 1 (required) specifies the data set key looked for.

key-2 Parameter 2 (required) specifies the new data set key.

filename Parameter 3 (optional) specifies the file name of key-1 to be changed to.

EXAMPLE

RENAME 2,@2

The key portion of the directory identified by 2 is replaced with a new key of CAN-code "2" -19136).

RESTORE

Regenerates the Data Saved

The RESTORE directive will cause data appearing on "filename" in a special archival storage format to be regenerated. "Filename" can be assigned to any valid input device including unit record devices. The information on "filename" must have been produced by an earlier SAVE directive.

The data actually written by RESTORE is determined by the following:

- . the data originally saved
- . any differences which can exist between file assignments at the time the data were saved and at the time the RESTORE is executed
- . the current contents of the user's directory on RAD or the current absence of a directory (RAD assigned to NO)
- . the presence or absence of the "directory-suppression-indicator."

Normally, all data written by a single SAVE will be restored by a single RESTORE. Regeneration will occur using the current file assignments. Thus, data can be regenerated on devices different from those originally containing the data. The new devices should not have a device size smaller than the size of the records saved. If the FORTRAN Direct Access Routines are to be used to manipulate the recreated data, the sector size of the partition receiving the regenerated data must match that of the partition from which the data were originally taken.

In addition to saving and restoring data sets, these directives can save and restore the current RAD directory and the current contents of the DEFINE table, which is discussed later. It is possible to use SAVE and RESTORE to reorganize the arrangement of data sets on sub-partitions. First, the data sets are saved then, the directory is regenerated for the new arrangement and the data sets are restored to the new locations defined for them.

Data sets will always be restored to the locations currently defined for them in the user's directory on RAD, even if some or all of the data sets formerly stored on a single file are now allocated to different files. If any data set does not appear in the current directory (or if RAD is assigned to NO) the following error message is output to the CO file.

DIRECTORY ENTRY FOR DATA SET XXX (ZZZZZZ) ON FILE YYY MISSING DATA SET WILL NOT BE RESTORED

Data sets formerly occupying entire partitions can be restored to sub-partitioned files, although frequently, it will be necessary to truncate excess data by ignoring any FILE TOO BIG error message.

SYNTAX

RES[TORE] filename[,directory-suppression-indicator]

filename	Parameter 1 (required) specifies the file name to be regenerated.
directory-suppression-indicator	Parameter 2 (optional) specifies any parameter that will be identified as such by the executive text scanning services. If the "directory-suppression-indicator" is present, any directory or DEFINE table appearing in save format on "filename" will be ignored. If the parameter is missing, any directory on "filename" will be written in the initial sectors of the partition to which RAD is currently assigned, and any DEFINE table will replace the DEFINE table in core.

SAVE

Causes User's Directory on RAD, Current Core DEFINE Table and/or Datasets to be Written on a File in a Format for Archival Storage

The SAVE directive causes the user's directory on file RAD and/or the current core DEFINE table and/or one or more data sets to be written on "filename-2" in a special format suitable for archival storage. The items so saved can be regenerated at some later time by the RESTORE directive. The items actually saved are determined by the first parameter.

SAVE and RESTORE are used as a complementary pair which can be used to preserve disc or magnetic tape files on some alternate medium, such as a different disc or tape. Magnetic tape files containing variable length records can be saved without restriction so long as the record will fit into core allocated to DAMP. The restrictions for COPY, DISPLAY and SET do not apply to SAVE. Careful use of the many options applicable to SAVE will allow practically any part of any partition or any input file on any device to be written on a system file for later regeneration.

If the "sequential-indicator" is omitted, an attempt is made to locate "key" in the user's directory on the current RAD file. If it is found, the data set defined by the directory is saved on "filename-2." If "key" is not in the directory, the directory is searched a second time to determine the existence of any entries describing data sets residing on "filename-1." If any sets are found during this search, they are saved on "filename-2"; unoccupied portions of "filename-1" are ignored. If, however, there are no references to filename-1 in the directory (or if RAD is assigned to NO), the DEFINE table is searched for data set key corresponding to the item entered as "key." If "key" does not appear in the table, the entire partition to which "filename-1" is currently assigned will be saved. If "key" does appear in the DEFINE table, DAMP saves only those sectors of the partition currently assigned to "filename-1" which could be used by the defined data set. Refer to the DEFINE directive.

Note: "filename-2" can be assigned to any valid output device.

Undetectable errors can occur if any SAVE directive causes DAMP to access portions of a partition which contain records which were not written under operating system control or which were written at the end of the current partition by routines which were actually accessing a longer overlapping partition. The system will drop excess data from any physical record which will not fit into the core allocated to DAMP. To avoid data loss, the user should ensure that DAMP is allocated enough core to contain its code, its data areas, and a buffer a few words longer than the longest record to be processed.

Sequential files containing records with an odd number of bytes should not be saved by the SAVE directive. If they are, records containing an odd number of bytes will be saved with an additional byte appended to them.

SYNTAX

```
SAV[E] [DIRECTORY ],filename-2[,sequential-indicator]
      [DEFINE-TABLE]
      [EVERYTHING]
      [key]
      [filename-1]
```

Parameter 1 (required) specifies one of the following:

- | | |
|----------------------|--|
| DEFINE-TABLE | specifies the current contents of the DEFINE table will be written on "filename-2" in the save format. |
| DIRECTORY | specifies that the user's directory on the current RAD file will be written on "filename-2" in the save format. |
| EVERYTHING | If DEFINE-TABLE, EVERYTHING, or DIRECTORY is selected for the first parameter, the "sequential-indicator" is ignored. Furthermore, for EVERYTHING or DIRECTORY, RAD must be assigned to a disc partition and must contain a valid directory. If EVERYTHING is specified and several directory entries describe overlapping data sets, the duplicated information will be saved each time a new directory entry references it. This is also true when a single file containing overlapping sets is being saved. |
| key | specifies a data set name. |
| filename-1 | specifies a filename. |
| filename-2 | Parameter 2 (required) specifies the filename where parameter 1 is written to. |
| sequential-indicator | Parameter 3 (optional) specifies any parameter that will be identified as such by the executive text scanning services. |

Since the syntax of "key" and of "filename-1" are substantially identical, if either of these forms appears it is scanned as one entity and interpreted as one or the other depending on several factors.

EXAMPLE

```
SAVE LB,SO,SEQ
```

Sequential file LB, starting at its current position, is saved on file SO, assuming that LB is terminated by a double file mark.

SET

Places a Specified Value Into Every Word of a Record or Sequence of Records in a Data Set.

Since a data set can be an entire partition, SET is very useful for initializing a partition before a data set is ever written on it. By clearing an entire partition to some constant value, unwritten trailing sectors or extraneous file marks, both of which can adversely affect COPY or SAVE, can be easily eliminated. Although it is somewhat cumbersome, SET can also be used to change single words on a disc file by using one SET for each word to be changed.

The SET directive writes the value specified by the fourth parameter into every 16-bit word of the designated records of the data set specified by "key." A total of "integer-2" records will be modified, starting with record number "integer-1." If the fourth parameter is some "alphanumeric-literal," it will be left justified, trailing spaces will be added, and the 16-bit value used will be the MODCOMP ASCII representation of the first two characters of the result. If the fourth parameter is an "empty-parameter," the 16-bit value will be hexadecimal 2020. If the fourth parameter is omitted, the value specified will be zero. If "integer-1" is omitted, the first record filled will be record 1. If "integer-2" is omitted, all records from "integer-1" through the end of the data set will be modified. The amount of information changed by SET is influenced by the same factors which affect DISPLAY. Refer to section 4.3.

SYNTAX

```
SET key [,integer-1][,integer-2][,integer-3]
                                     [,alphanumeric-literal]
                                     [,empty parameter]
```

key	Parameter 1 (required) specifies the data set name with the records that will be modified.
integer-1	Parameter 2 (optional) specifies the record number of the first record to be modified. The default is one.
integer-2	Parameter 3 (optional) specifies the number of records to be modified.
	Parameter 2 (optional) specifies one of the following:
	Parameter 3 (optional) specifies one of the following:
integer-3	specifies a signed or unsigned integer.
alphanumeric-literal	specifies any parameter which the COLLECT service will classify as a word parameter.
empty-parameter	is indicated by placing a non-space delimiter or space(s) followed by a non-space delimiter immediately after the delimiter for "integer-2."

EXAMPLES

SET BI 1 6 -1
SET 12,,, #A5
SET @HAT 26 3,,
SET SC

TWIN

Places a New Entry in the User's Directory on RAD

The TWIN directive places a new entry in the user's directory on RAD. The file name and sector limits of this entry are duplicates of the file name and limits for the data set identified by "key-1." The key of this new entry is "key-2." This directive does not affect the DEFINE table.

SYNTAX

TWI[N] key-1,key-2

key-1 Parameter 1 (required) specifies the data set key to be duplicated.

key-2 Parameter 2 (required) specifies the data set key of the new entry.

EXAMPLES

```
TWIN 52 787
TWI BO SC
TWIN @5.M,370
```

A new entry is placed in the directory. The new entry's key is 370 and its three other words match those of the entry for key @5.M (equivalent key: -12803).

Prints Output from DISPLAY Directive 40 Characters Per Line

The TWO directive causes DISPLAY to print its output in 40-character lines. It is primarily useful when output is directed (by assigning LO) to a narrow device such as the teletype. ONE is in effect by default (refer to the ONE directive).

A	40 characters
H	8 hexadecimal words
I	2 numbers
R	2 numbers
L	2 numbers
D	2 numbers
E	2 numbers

SYNTAX

TWO

(

l

(

l

l

l

CHAPTER 4 GUIDE TO THE USE OF DAMP

4.1 FILE USAGE

DAMP directs most operations involving I/O devices to files named by the user. Usually, the user selects these names and assigns them as necessary. In addition to user files, DAMP uses the following file names for the indicated functions. Except for RAD, the files are accessed sequentially.

DAMP will read directives from file AI if the AO option bit is on and from CI if it is off. The other options are not referenced by DAMP and can be set as desired.

If a directive error is detected and CI is not a terminal device, then DAMP will set the system GO-NOGO flag. If DAMP is loaded with Job Control, the flag will be interrogated by Job Control both before and after DAMP runs.

<u>FILE NAME</u>	<u>FUNCTION</u>
AI	Directives are read from this file if the AO option is on when DAMP is initially loaded.
CI	Directives are read from this file if the AO option is off when DAMP is initially loaded.
CO	ACTION requests are written on this file (using the executive message service) if it is not on the same device as CI (AI). Error messages requiring operator intervention are written on this file (again using the message service). An initiation message is written on this file if it is assigned to the same device as CI (AI). Error messages are written on this file if it is assigned to the same device as CI (AI) but it is not assigned to the same device as LO. (These messages are written directly without using the message service.)
LO	Error messages are written on this file. All directives are listed on this file if it is assigned to a device different from the CI (AI) device. The reports generated by the LIST and DISPLAY directives are written on this file.
RAD	The current direct access data set directory is read from this file. Any modifications of or replacements for the directory are written on this file. RAD must be assigned to a direct access device.

4.2 KEYS

A particular data set in a file is specified by a unique key which can be regarded as the name of that data set. This key can be a positive decimal number. If it is in the range $1 < \text{key} < 999$, it can be used as the direct access logical unit number in FORTRAN programs. It can be more convenient to use other forms for the key. DAMP will process keys in the following formats. For each format a corresponding canned file name format is also defined. This canned name is used to reference the data set if the set does not appear in the user's directory on RAD.

- The key can be a decimal number in the range $1 < \text{key} < 32767$. The canned file name corresponding to this key will be the CAN representation of the number left-justified with trailing blanks. Therefore, for key 1 the file name is canned "one" "space" "space," and for key 32767 the name is canned "three" "two" "seven."
- The key can consist of the character "@" followed by at least three legal CAN characters, followed by characters which will be ignored until the next delimiter. The key will be the three characters following the "@," converted to CAN code. The canned file name will be the same as the key.

NOTE: Leading or embedded spaces are valid in this format.

- The key can consist of a letter (A to Z) or a colon (:) followed by any valid CAN character. The key will consist of the two characters in ASCII. The file name will consist of the CAN representation of these two characters with a trailing space or of the CAN representation of the two characters followed by the third character in the parameter if the parameter is longer than two characters. The space is a delimiter in this format.

NOTE: Each of the above formats generates a 16-bit binary key. Care must be taken to prevent accidental duplication of keys. For example, keys EX and @KC5 are equivalent. The final binary value for a key can not be zero.

4.3 REFERENCING DATA IN A DATA SET

Data in a data set is referenced by the directives COPY, SET, and DISPLAY. The record sizes, the number of records involved, and the starting positions are determined by the type of device containing the file, whether the data set is in the current DEFINE table, and whether the data set is in the user's directory on file RAD. First, the following three parameters are established.

<u>PARAMETERS</u>	<u>DATA SET IN DIRECTORY</u>	<u>DATA SET NOT IN DIRECTORY</u>
INITIAL SECTOR LIMIT	From Directory	0
FINAL SECTOR LIMIT	From Directory	32,767
ACTUAL FILE NAME	From Directory	CAN name associated with key (Refer to 4.2.1)

Next, the physical record size for this data set is determined from the MAX IV and MAX 32 executives using the file name ascertained above. If the file has an unlimited record size, the physical record size is set to 128 words. Now the following two parameters are derived for this data set.

<u>PARAMETERS</u>	<u>DATA SET IN DEFINE TABLE</u>	<u>DATA SET NOT IN DEFINE TABLE</u>	
		Disc	Not Disc
LOGICAL RECORD SIZE	From table	Physical Record	Physical Record

MAXIMUM RECORD NUMBER	From table	Number of sectors in partition	32,767
--------------------------	------------	--------------------------------------	--------

Once the above parameters have been ascertained, the transmissions involving the data set proceed as follows:

<u>PARAMETERS</u>	<u>FILE ON DISC</u>	<u>FILE NOT ON DISC</u>
PHYSICAL RECORD MODIFICATION	None	Physical size set to logical size.
RECORD DERIVATION	Logical records are extracted from or inserted into disc sectors using the parameters deter- mined above.	One physical record is transmitted for each logical record. No size checks are made and no extractions or insertions are done.
STARTING POSITION	Logical record 1 is first transmitted for COPY and logical record "integer-1" for SET or DISPLAY. Displacements are calculated from the parameters deter- mined above.	The next sequential record on the file is assumed to be record 1 for COPY or record "integer-1" for DISPLAY or SET.
TRANSMISSION TERMINATION CONDITIONS	(1) Input EOF on last sector used in attempted extrac- tion of requested record. (2) Final the sector limit exceeded. A par- tial logical record can have been transmitted. (3) and (4) same as (2) and (3) for non- disc files.	(1) Input EOF (2) The number of records requested has been transferred. (3) The record with maximum record number for this file has been transmitted.

4.4 CHAINED RAD PARTITIONS

The main task performed by DAMP is to insert, as directed, the indicators for a chained RAD partition. DAMP does **NOT** use chained RAD partitions; it operates only on the RAD format partition currently assigned to RAD.

4.4.1 General Information

The chained RAD indicators are created by using the ENTER command for a 'special' dataset

with a name in the range "@\$\$ " to "@\$\$\$". In more general, terms the data set name is the CAN code of \$\$ character, where character is any one of the CAN codeable characters:

space, A-Z, 0-9, : (colon), . (period), \$ (dollar)

This allows up to 40 chained entries per directory. The fact that the CAN code characters provide a collating sequence (i.e. space, A-Z, 0-9, :, \$) is used by the FORTRAN Run-Time Library to control the order of searching.

The ENTER command associates a logical file name with the data set entered. The same is true for the @\$\$ entry. The sector limits are not relevant for these entries and are therefore set to zero.

Therefore:

```
ENTER @$$A BI
ENTER @$$L LNK
```

will make two directory entries for chained RAD. The command

```
ENTER @$$R RAD
```

is illogical since the current RAD partition is being chained to itself; repeatedly; this is errored with FILE TOO BIG.

Application of commands to the @\$\$ data sets is restricted. The only commands applicable are:

```
REMOVE @$$ and
RENAME @$$x @$$y
```

RENAME will allow @\$\$ data sets to be called by another @\$\$ name but will not allow a 'normal' data set to change into an @\$\$ set, or the reverse and therefore, the following are invalid.

```
RENAME @$$x xxx and
RENAME xxx @$$x
```

The REFILE command does not distinguish between data set types and so will REFILE all entries referenced by the specified command.

SAVE operates in a simple manner. The directory entries are copied regardless of type. SAVE EVERY omits the copy of @\$\$ dataset contents, nor can an @\$\$ entry be saved directly. RESTORE obviously reverses the process. Since @\$\$ datasets are pointers only they have no content to "save".

This mechanism ensures that each RAD format partition and its associated data sets form a self-contained unit. The chaining mechanism then interlinks a series of such units.

4.4.2 FORTRAN Mechanism

The mechanism used to handle multiple RAD directory searches is described in diagram form on the following page. Only two specific rules apply:

- 1) RAD must be assigned to the first directory to search (no change from the previous state)
- 2) All logical files associated with @\$\$ entries must be assigned to RAD format files, or to NO

The question of performance requires comment. Previous versions searched the only RAD directory on the first access to the data; the procedure RERAD could be called by the program to force the appearance of a "first access" on the next access. This remains unchanged. Therefore, any overhead imposed by chained RAD operations is constrained to the first access of the data set (or simulated first access, if RERAD is used).

The time taken to locate the required data set increases in proportion to the number and size of directories to be searched. The overhead on a RAD directory which does not use chained entries is very small.

The following figure, Figure 4.1, illustrates the logic of the FORTRAN run-time data set location.

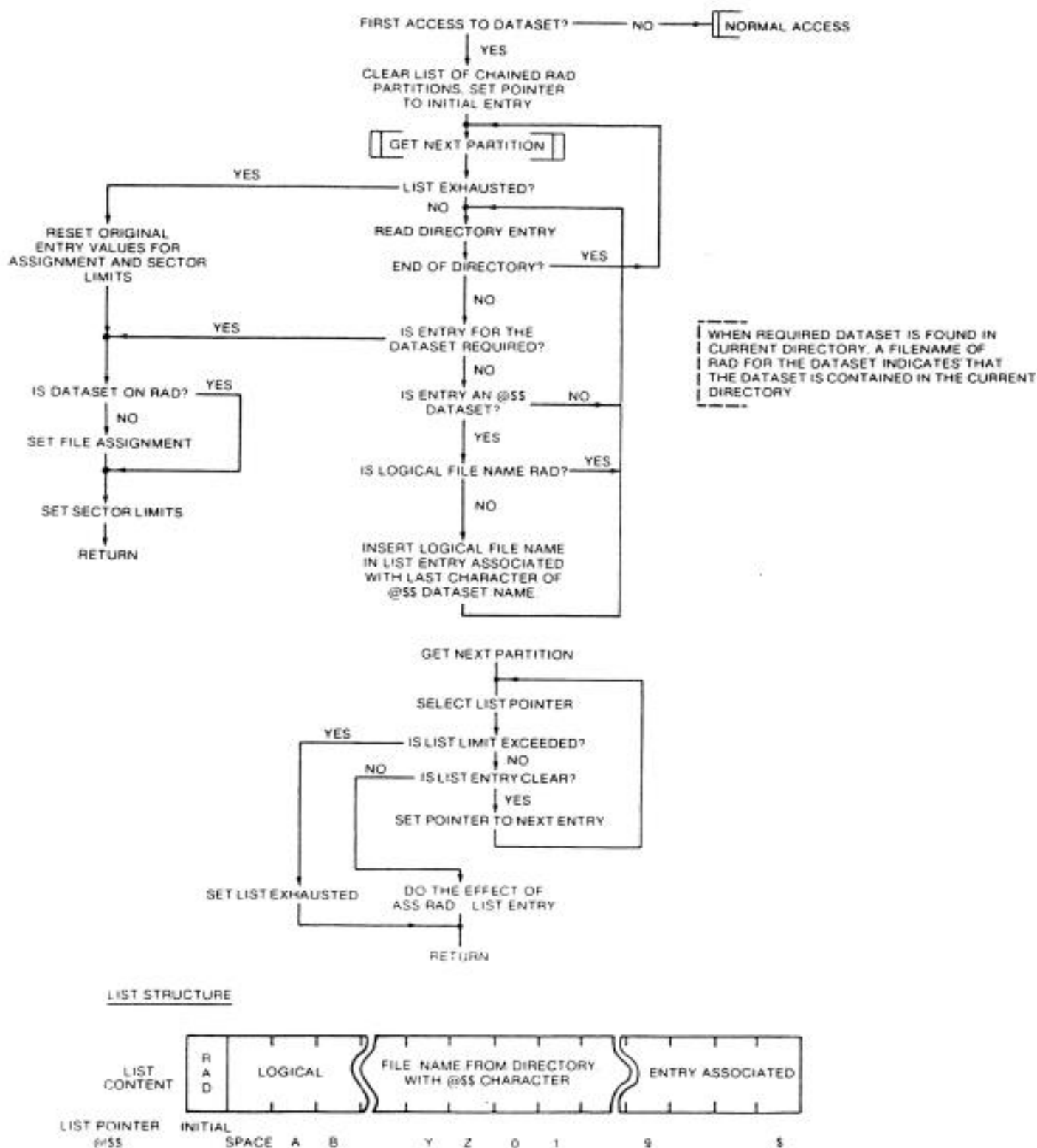


Figure 4.1 FORTRAN Run-time Data Set Location

4.4.3 Usage

Assume a set of RAD format partitions called A, B, C and D with the following structures:

A contains	100	BO	0	999
	101	RAD	100	299
B contains	200	BI	0	999
	201	RAD	100	399
C contains	300	SI	0	999
	301	RAD	100	499
D contains	400	SO	0	999
	401	RAD	100	599

Assume also a set of FORTRAN (or any other, using the FORTRAN Run-Time Library Random Access routines) programs called:

FORT1 using datasets 100 and 101
FORT2 using datasets 200 and 201
FORT3 using datasets 300 and 301
FORT4 using datasets 400 and 401

Because the chained RAD facility requires the new version of the Run-time libraries (EDIT, M4EDIT), the programs FORT1 and FORT2 are recompiled and linked with the relevant version of the Run-Time Library. (If the unlinked form of the program is not available a re-compilation is needed.)

4.4.4 Use of RERAD

The library procedure RERAD is available via a CALL statement in the program. Its effect is to change various markers which make the next access to the specified data set appear to be the first. Therefore, the chained search is entered for the data set.

This procedure is used when the program suspects that a change has been made to the directory which could influence its operation. The method of detection is program dependent. Operator notification via sense switches and task variables are common mechanisms.

4.5 SYSTEM CONFIGURATIONS

DAMP is most valuable on a system containing at least one disc drive. It is generally helpful to have one partition available for scratch use. Files on magnetic tape can usually be processed so long as they contain fixed length records. Both the number of records on the file and the fixed length of each record must be known if files on non-DMP devices are to be processed. This information can usually be obtained by processing the file twice.

If a disc is not available on the system upon which DAMP is being run, the directives referencing direct access directories cannot be used. Magnetic tape files can be manipulated with other directives as can files on non-rewindable devices, so long as their characteristics are known.

DAMP is not particularly useful on systems with no rewindable devices. It can be used to copy arbitrarily formatted files, to display such files, or to generate files containing constant information. Such files can also be recopied in the archival format generated by SAVE, but with little benefit. Most of these functions can be performed more conveniently with other MODCOMP software.

4.6 SPECIAL DATA FORMATS

DAMP uses the executive's byte-string scanning services to interpret its directives. With the following exception, the names of the directives and their parameters must adhere to the syntax allowed by these services. One special character, @, is valid as the first non-blank character in a parameter which designates a data set key. Recognition of this character allows a more specialized format for keys (refer to 4.2).

The most important data structure used or created by DAMP is the direct access directory. This directory will be written onto, or must already be recorded on, the first sectors of the partition to which the user has assigned the file name RAD. Each directory sector is laid out in the following manner.

- Word 0, bits 1 to 15 - The count of the number of entries in this sector (N).
- Word 0, bit 0 - The directory continuation flag. If it is set to 1, the directory is continued on the next sector. If it is set to 0, this is the last (or only) directory sector.
- Words 1 to 4N - The directory entries.

Each entry consists of four words giving the following information:

- Word 0 - The directory key.
- Word 1 - The canned file name which has been assigned to the partition actually containing the direct-access file.
- Word 2 - The sector displacement of this file within the partition that is, the relative sector address of the sector containing the first part of the first record of the file.
- Word 3 - The sector limit for this file; that is, the relative sector address of the sector containing the last part of the last record of the file.

The directory key for FORTRAN direct access files is the logical unit number for that file. Empty entries are indicated by zero keys and file names. The other two words are ignored.

NOTE: The above format allows the introduction of "empty" sectors into the directory. If the first sector of the partition, or any sector following a directory sector with the continuation bit on, has zeros in bits 1 to 15 of its first word, the remainder of its words will be ignored for normal directory purposes.

The continuation bit of an empty sector functions normally. A user can keep any desired information in the body of these sectors.

The user should of course be aware that embedded empty sectors slow down directory searching. DAMP places empty sectors into the directory whenever the APPEND directive is issued. The second word of these sectors contains the file name of the file originally

containing the directory being appended. This is used by the ELIMINATE directive. Any programmer including empty sectors in his/her directory should either avoid using APPEND or should always zero the second word of his/her "empty" sectors.

The DEFINE table created by DAMP consists of up to ten 3-word entries. Word 0 of each entry is the key of the set being defined. Word 1 is the number of records in that set. Word 2 is the number of words per record. A zero key designates the end of the table; one zero word is stored after the table to terminate a full table.

The format of most listings produced by DAMP is self-explanatory. Record size in words and individual record numbers are indicated by DISPLAY. LIST produces information about the sector arrangement of the directory as well as designations for empty entries, sequences of empty entries, and empty sectors.

4.7 SAVE/RESTORE ARCHIVAL FORMAT

A series of records followed by a file mark is generated for each data set processed by a SAVE directive. An additional file mark is written after the last data set saved. The processing of a RESTORE directive will cease when this file mark is encountered.

All of the records generated by SAVE are in the MODCOMP standard binary format. The file generated for a given data set consists of a header record followed by data records. The header record has the following format:

WORDS	CONTENTS
-------	----------

- | | |
|-----|--|
| 0 | #0300, the standard binary identifier. |
| 1 | 80, the record byte count. |
| 2-3 | The characters DAMP in ASCII. |
| 4 | The data set key if this is a random save, the file name (word 5) if this is a sequential save, or zero if this is a directory save. |
| 5 | The name of the file from which the subsequent data were taken, or zero if this is a DEFINE table save. |
| 6 | The number of the first physical record saved. For a random save of a data set from the directory, this is word 2 of the directory entry. For a random save of a data set not in the directory or for a save of the directory itself, this is zero. This word is not used for a DEFINE table save. For a sequential save, this is the file position index from the operating system's file assign list for the file named in word 5. |
| 7 | The number of the last physical record to be saved. For a random save of a data listed in the user's directory this is word 3 of the directory entry. For a random save of a data set not listed in the directory but listed in the DEFINE table, this is a number calculated by applying the ALLOCATE algorithm to the DEFINE data while assuming that the initial sector allocated is numbered zero. For a random save of data set listed neither in the directory nor in the DEFINE table, this is one less than the number of sectors specified for this file's partition in the operating system's physical device tables. For a directory save this word |

is zero, and for a sequential save it is minus one. It is not used for a DEFINE table save.

- 8 Unused.
- 9-39 For saves of the DEFINE table, these words contain the table with a trailing zero. No data records are generated for saving the DEFINE table, since it is completely contained in the header record. The data records generated by other types of saves have the following format:

The data records have the following format.

WORDS	CONTENTS
0	#030 or #07 is a standard binary indicator.
1	The physical device size, in bytes, from the operating system's physical device table corresponding to "file-name-2" from the SAVE directive. If the device size is unlimited the value used is 256.
2	A record checksum; the two's complement of the sum of all the words in the record except word two.
3toN	The data actually being saved, written in an encoded format. N is one less than one-half of the value in word one.

The data being saved, whether they are from a directory, a data set, or a sequential file, are encoded simply by prefixing them with a control word. This word contains the size in words of the resulting record and is calculated by the formula $(UBC+3)/2$, where UBC is the byte count obtained from the operating systems for the original record. File marks are encoded as a record of length one. When the save of a particular item is complete, an inconsistently encoded record with a length of one but a value of zero is generated. This record must be detected for a RESTORE to function properly.

The encoded records are packed into the N minus 2 available words of the data records. Reblocking is performed as required. If the final encoded record does not fill a data record, the remaining words of the record will be ignored except for the purpose of checksum calculation.

4.8 LIMITATIONS

DAMP was designed for operation with files which can be read properly in the MODCOMP nonstandard binary mode. In order to process other files, DAMP must be given the actual number of records appearing on those files. The storage file produced by SAVE and interpreted by RESTORE is in standard binary and so can be freely directed to non-DMP devices.

DAMP was designed for use with partitions limited to a maximum of 32K sectors or records. This limitation is fixed in the directory format by the use of 16-bit sector numbers. Similarly record numbers are also fixed. When encountering a partition exceeding this limit, DAMP truncates the size to 32K. This can present various problems when using this type of partition.

The absence of a data set record size in the directory provides a flexible method of accessing data. It can also cause problems. The user must ensure that all FORTRAN programs manipulating the data records use comparable record descriptions in the DEFINE FILE statement. Records do not contain a record terminator.

DAMP does not perform any validity checking on the contents of purported directories. If the name RAD is assigned to a file which does not contain a valid directory at a time when a directory is required, DAMP will usually have to be aborted by the operator. Similar considerations apply to the Run-Time Library; this is a significant factor in the use of chained RAD partitions.

Although DAMP can process the physical records produced by L\$CMW, the source compression routine, it cannot interpret the images contained within these records. Compressed files must be copied, displayed, or otherwise manipulated on the basis of the actual physical records contained in them.

The COMPRESS directive, which is used to recover sub-partitions which are no longer being used, operates on a single file. Power failures or other system failures during the file reorganization can cause loss of data. If provision must be made for this contingency, a SAVE should be performed on the file before it is compressed. During an entire COMPRESS operation, the structure of the file being compressed varies in such a way that other programs operating in the multiprogrammed environment cannot reliably access the file. If a file used by many concurrent programs must be compressed, you must first take exclusive use of the file on behalf of the task which is running DAMP then, give up exclusive use after the compression is complete. Upon completion of the COMPRESS operation, tasks which use the direct access files are deestablished and then reestablished in order to update the direct access pointers of the programs resident in core at the time of the COMPRESS.

DAMP determines whether the directory on RAD can be extended by searching the directory for references to RAD. If other file names are also assigned to RAD's partition, their data sets will not be similarly protected.

APPENDIX A ERROR MESSAGES

DAMP checks for errors and issues appropriate error messages. The error messages are written on file LO. Some of these errors cause the remainder of the operation of the current directive to be bypassed while others cause DAMP to be placed in the HOLD state until action is taken.

DAMP issues the following error messages.

BOUNDARY ERROR

An access has been made outside the limits of a data set during a COPY, DISPLAY or SET. This is usually caused by an improper specification in the associated DEFINE directive(refer to Chapter 3 Directives). If the data sets do not appear in the user's directory, DAMP can be aborted because of partition boundary violations or can write off the end of a magnetic tape.

CHECKSUM ERROR

This is part of a HOLD message written on OC. The checksum test on a record from the file specified in a RESTORE directive has failed. If execution is resumed, DAMP will accept the error record as if it were correct.

COMMAND ERROR

Indicates the directive just entered is syntactically incorrect. This message appears if the text manipulation executive services take an error exit or if a specified key or file name has the value zero.

CORE EXCEEDED

Any of the following conditions can exist:

- . A logical record to be processed by COPY, DISPLAY, SET or RESTORE is too long to fit in the core available to DAMP.
- . The amount of extra core available to DAMP is too small to accommodate a temporary directory being built by the COMPRESS directive processor.

DIRECTORY FULL

Any of the following conditions can exist:

- . For an APPEND directive, the directory on "filename" would overlay files sharing the partition with RAD if it were added to RAD. This message can be generated erroneously if the "filename" directory contains references to a RAD (such as itself) different from the current RAD.

- For an ALLOCATE, ENTER, or TWIN directive, the RAD directory must be expanded to accommodate the new entry but there are data sets also residing on RAD so that it cannot be extended. Try compressing RAD to recover any wasted space.

NOTE: If RAD does not also contain data sets and if the directory is extended past the end of the RAD partition, DAMP will be aborted.

DUPLICATION KEY

There is already a data set in the user's directory on RAD with the same key as the new key specified in ALLOCATE, ENTER, RENAME, or TWIN.

FILE TOO BIG

Any of the following conditions can exist:

- The directory specified in CREATE will overflow the end of the partition containing RAD.
- The newly allocated data set will not fit in the remaining unused space of "filename." Try compressing "filename." If "filename" is RAD, also try purging RAD.
- The newly allocated data set being created overlapping another data set extends from the beginning of the second data set past the end of the disc partition. It is valid to overlap several data sets, but a data set can not extend past the end of its file.
- The data set specified in ENTER overlaps the RAD directory or has a negative low-sector address.
- If the message is part of a HOLD message on OC, the RESTORE directive has detected a data set that requires more space than is currently allocated to it. If DAMP is allowed to continue, the data set will be restored until its space is filled, and the excess data will be bypassed. The data set name is given in decimal and CAN-code.

FUNCTION ERROR

Any of the following conditions can exist:

- The file name in an APPEND directive is RAD.
- The high value on an ENTER directive is less than the low value.
- The number of data sets on "filename" after a COMPRESS directive was executed is different from the number before the compression was performed. Either the assignment of RAD has been changed or RAD was improperly assigned to a partition which overlaps the partition containing the compressed file, so that the directory has been destroyed.

- Extraneous file marks appear in, or a header record is missing from, the file specified in a RESTORE directive.
- The current SAVE directive specifies DIRECTORY or EVERYTHING, but there is not a valid directory on RAD.
- An ACTION directive was entered from the system CO device.

INVALID SECOND KEY

The second key in the current ALLOCATE directive cannot be found in the user's directory on RAD or else designates a data set residing in a file different from the new data set's "filename."

KEY NOT FOUND

The key specified in REMOVE or the first key specified in RENAME or TWIN does not appear in the user's directory on RAD.

NO SUCH FILE

Any of the following conditions can exist:

- The file name specified in ELIMINATE does not appear as a flag in any empty sector in the current RAD directory.
- The current directory does not contain any data sets residing in the file specified on the COMPRESS directive just issued.

END OF MEDIA ON XXX

A SAVE or RESTORE directive found an end-of-file condition on file-name XXX when assigned to a magnetic tape. A console interrupt followed by /RES indicates that a continuation magnetic tape has been mounted.

NOTE: You are responsible for ensuring that the tape sequence for RESTORE is identical to that used for the SAVE.

DIRECTORY ENTRY FOR DATA SET XXX (ZZZZZZ) ON FILE YYY MISSING DATA SET WILL NOT BE RESTORED

This message is given when a RESTORE directive finds a missing directory entry for a data set. It will not be restored!

HAVE YOU SAVED THE FILE AND ARE THE WRITE PROTECT SWITCHES OFF?

Before a COMPRESS is performed this message will appear reminding the user to verify that a backup should be made due to possible hardware/electrical malfunction.

NEXT COPY WILL BE ASCII FOR 'FROM' FILE

This is an informational message indicating that the ASCII directive was used.

NEXT COPY WILL BE BINARY FOR 'FROM' FILE

After COPY with ASCII is finished, further COPY directives will be in binary unless an ASCII directive precedes it.

XXX DIR. CORRUPT? S=#\$\$\$ H=#hhh M=#zzz

This message, indicating a possible directory corruption, forces a COMMAND ERROR. The message contains four items of variable data, XXX being the name of the file on which the potential directory corruption has occurred. The remaining variables are output as hexadecimal values:

S=sector number in directory ($0 \leq s \leq \text{directory size}$)

H=first word (0) of this directory sector including bit 0

M=maximum number of entries expected in this sector

The error is reported when $H \neq M$. This can occur in two cases:

- There is a genuine corruption of the directory.
- The directory was created with a 'record size of A' but its current usage has a size B which is less than A, i.e. a discrepancy in the directory usage has occurred. (Refer to Special Data Formats section.)

DATASET \$\$ TYPE INVALID IN THIS COMMAND

The dataset names @\$\$space to @\$\$\$ are used to indicate chained RAD partitions. An attempt to use an invalid command on a data set of this type has been made.

WARNING: XXX TRUNCATED TO 32K SECTORS

The format of Random Access Directories will only contain a maximum of 32K sectors. This message reports that the file XXX has a size which exceeds this limit, and only the first 32K will be manipulated.

XXX HAS UNLIMITED RECORD/SECTOR SIZE

This is an information message to indicate that the current file has no restrictions on size. DAMP assumes a default of 256 bytes.

INVALID/UNCOMPILED FORMAT INDICATOR

This message is given by the DISPLAY command as a termination of the command whenever the format indicator specified is unrecognized. The main cause of this, apart from genuine errors, is requesting a display option which has not been compiled into DAMP.

APPENDIX B EXAMPLES FOR USING DAMP

B.1 USE OF DIRECTIVES

Many of these examples show both Job Control directives and directives for other system processors such as DAMP. The appearance of a \$JOB directive below is not meant to imply that the directives which follow it cannot be embedded in a longer job stream. Rather, it serves to indicate that the default file assignments and default system option settings should be assumed to be in effect before the next directive is interpreted.

In the following, examples actual partition names will be indicated by the letters PNC where the variable c is included to allow the specification of several different partitions. Other device names will be defined as they are used. It is also assumed that a disc exists and that the system includes default assignments of the following names to disc partitions.

```
SC
SCA
SCB
```

It is further assumed that these files will not conflict with the partition names used in the examples, unless, it is explicitly indicated that these file names are to coincide with specific partitions.

B.1.1 Defining Simple Direct Access Scratch Files

A FORTRAN programmer wrote and compiled a program named ALPHA. In it he defined three direct access files as follows:

```
DEFINE FILE 10(50,60,L,IV1)
DEFINE FILE 65(100,75,E,IV2)
DEFINE FILE 122(1500,125,U,KL7)
```

All of these scratch files are used internally by this one program. The program itself has been placed in unedited form on the system library which resides on file LB. Normally, the following Job Control sequence might be used to execute this program.

```
$JOB
$ASSIGN RAD=NO, 10=SCA, 65=SC, 122=SCB
$EXECUTE ALPHA,LB,LINK,NOMAP
```

If other file assignments were needed, they would be added to the \$ASSIGN statement. Now suppose that you determine that both data sets 10 and 65 will fit on the partition to which SC is assigned with at least one sector left over, and that sharing the partition will thus materially reduce the execution time of your program, even when the extra processing time for DAMP is counted. The following sequence can now be used to execute your program.

```
$JOB
$EXECUTE DAMP
ASSIGN RAD=SC,122=SCB
CREATE 1
```

```

ALLOCATE RAD,10,50,60,L
ALLOCATE RAD,65,100,75,E
EXIT
$EXECUTE ALPHA,LB,LINK,NOMAP

```

If additional file assignments are needed, a \$ASSIGN statement containing them can be added before either \$EXECUTE statement.

B.1.2 Initializing a New Direct Access File

Partition PNR has been set aside for a programmer's permanent direct access file which is to contain ASCII data. Before he begins to use the file, every record should be set to all spaces. It seems wasteful to write a program for this one-time job. The following DAMP directives will produce the desired result.

```

ASSIGN RAD=NO, BF=PNR
SET BF,,,
EXIT

```

B.1.3 Patching Object Records

You desire the ability to read integers left justified in input fields. In order to accomplish this you plan to maintain your own copy of the system library, suitably modified. The simplest feasible modification is to change the FORTRAN Run-Time Package format routine to ignore spaces on integer input. Thus, for example, the value 45 could be placed in a 3-character field in any of the following ways (b represents the space):

```

b45
4b5
45b

```

The last example is effectively left-justified.

A listing of the Package is available, but the source is not. You do not wish to keypunch programs from your listing, so you first enter the following sequence of directives to Job Control, MAX IV/MAX 32 Library Update, and DAMP.

```

$JOB CHANGE PACKAGE
$EXECUTE LIB
ASSIGN BI LB BO SCA
REWIND BI BO
GET FRT:K
WEOF BO
EXIT
$EXECUTE DAMP
ASSIGN RAD=NO
DISPLAY BO,,,H

```

You then examine the display produced by DAMP (without exiting from the processor). From your assembly listing of the Package you know that the zero appearing the string of hexadecimal values 5, 22, 0, 35, 3B, 3, and 29 should be changed to hexadecimal 11. You can locate this sequence in the display listing. Since the display prints 16 words per line, it is a

simple matter to calculate that the zero in the sequence was the sixty-sixth word in the BO file. Assuming the first word was numbered 1, this word was number 66. Therefore, you now command DAMP as follows:

```
DEFINE BO,200,1,U
SET BO,66,1,#11
```

This modification produced the desired change but also invalidated the record checksum. From the display listing you ascertained that the checksum word contained hexadecimal 3814. Your next directives follow.

```
SET BO,3,1,#3803
EXI
$EXE LIB,,NOLO
ASS BI LB BO PP SI SCA
REW BI SI
REPLACE FRT:K
COPY EXIT
```

Device PP is the paper-tape punch. The final sequence above produced a paper tape with the desired modifications.

B.1.4 A Simple Way To Combine Directories

You have two permanent collections of data sets on two partitions PN1 and PN2. Each partition has its own directory. The directory on PN1 contains the following information:

10	@RAD	451	781
30	@RAD	272	450
20	@RAD	782	1023

The directory on PN2 contains this information:

10	@RAD	272	602
30	@RAD	845	1023
40	@RAD	2	271
20	@RAD	603	844

You have a large library of programs, some of which operate with one group of data sets and some of which operate with the other. A few of your programs operate on data sets 10, 20, and 30 of either group, since they contain similar data. Which group is actually used is determined by the assignment for RAD at the time a particular program is run. Now you have devised a new statistical program which you wish to run using both groups of data sets simultaneously. There are many ways in which this program could operate. For example, it could call various library subroutines to reassign RAD after the sets on one partition had been processed and to cause RAD to be searched again. A simpler approach would be to generate a third directory for use by the statistical program. Let GP1 be an arbitrary file name which the user will assign PN1 and GP2 be a name to be assigned to PN2. For the statistical program the data sets on the first partition will be called 110, 120, and 130 while those on the second will retain their original names. The simplest way to create the desired directory would be to use the following DAMP directives. It is assumed that the you initially do not know the exact contents of his directories.

```
ASS RAD PN1  
LIST
```

Now with the necessary directory information before you, you issue the following directives.

```
ASS RAD SCA GP2 PN2  
MOVE GP2  
REFILE RAD,GP2  
ENTER 110,GP1,451,781  
ENTER 120,GP1,782,1023  
ENTER 130,GP1,272,450
```

The desired directory is not on SCA. A method of saving this directory is described in B.1.6.

B.1.5 Another Way To Combine Directories

For illustrative purposes a more complicated way of producing a directory similar to the one created in the previous example will now be described. The following DAMP directive sequence is more elaborate than the sequence from the previous example, but it is also somewhat more general. Note that the actual directory contents are not needed.

```
ASS RAD SCA GP1 PN2 GP2 PN2  
MOVE GP1  
RENAME 10,110,GP1  
RENAME 20,120,GP1  
RENAME 30,130,GP1  
ASSIGN RAD=SC  
MOVE GP2  
REFILE RAD,GP2  
ASS RAD SCA  
APPEND SC  
MOVE RAD,NO
```

NOTE: The above sequence generates a directory with the entries for file GP1 before the entries for file GP2, while the sequence in EXAMPLE 4 places the two groups in the other order. Either example could be adapted to produce a different ordering.

Refer to B.4 for using chained directories.

B.1.6 Saving Directories

Rather than regenerating the special directory created in either B.1.4 or B.1.5 each time it is needed, the user wishes to save it for later use. This can be done because he knows that the information used to create it will not be changed. The simplest way to save the directory is to use the SAVE directive. The device name PP represents the paper-tape punch. The following directives are entered after the directives in B.4.4 or B.4.5.

```
ASS BO PP  
SAVE DIRECTORY,BO
```

This sequence saves the directory off-line. To restore the directory whenever the statistical program is to be run, the following DAMP directives could be issued.

```
ASS RAD SCA BI PR
RESTORE BI
```

The device name PR represents the paper-tape reader. The directives restore the directory from the paper tape produced by the earlier SAVE directive.

Frequently it is more convenient to save information on the disc itself. File SCA is of course not available for permanent Storage. The user knows that there is ample space on his first partition for extra data, since the other partition contains a complete additional data set, set 40. (If he were to list his second directory he could tell that there is also one sector left over on his second partition.) The user decides to save his newly created composite directory on his first partition. He will call it the composite statistical directory, or CSD. The following directives are entered immediately after those that created this directory.

```
ASS RAD PN1
ALLOCATE RAD, @CSD,1,50,U
COPY SCA,@CSD
```

Whenever the user runs his statistical program, the following DAMP directives recover the directory.

```
ASS RAD PN1
COPY @CSD,SC
ASS RAD SCA
```

These instructions cause DAMP to print the message BOUNDARY ERROR. The message can be ignored or else the directive

```
DEFINE @CSD,1,50,U
```

Can be entered before the COPY directive.

B.2 PROCEDURES USING DAMP

In the following examples, actual partition names will be indicated by the letters PNc where the variable c is included to allow the specification of several different partitions. Other device names will be defined as they are used. It is assumed that a disc exists and that the system includes default assignments of the following names to disc partitions.

```
SC
SCA
SCB
```

It is also assumed that these files will not conflict with the partition names used in the examples unless it is explicitly indicated that these file names are to coincide with specific partitions.

B.2.1 Debugging Programs Without Endangering Data

A programmer has an elaborate system of programs which process information from a data base consisting of 47 different data sets. These data sets are stored on a removable disc

pack and are all described in a directory stored on partition PND of this pack.

All of the data sets contain unformatted data and reside in the same partition as the directory. The programs are stored on a smaller partition of the pack. A typical program accesses data from six or seven of the data sets and modifies one or two of the data sets on the basis of this information. The user frequently revises these programs and often writes one-time special-purpose programs to modify portions of the data. To debug these new programs, the user likes to work with a small subset of the files which are being changed and to display portions of the files before and after a trial run. Of course, the user wants the original files to remain unchanged by the test. Since the programs only rarely modify more than two of the data sets, the user devised the following procedure to simplify these debugging runs. It allows parameters for two data sets, although the second group of parameters are seldom needed. To use the procedure, the programmer first prepares the program either by compiling it or, if it is in object form, by mounting it on an appropriate device. Next the file name BI is assigned to the device containing the sequentially accessed test data (usually the card reader, teletype, or a magnetic tape) are also assigned at this time. Finally the user executes the procedure with the statement:

```
$DO TRIALR,K1,N1,S1,L1
```

or, if two data sets are to be modified, with the statement

```
$DO TRIALR,K1,N1,S1,L1,K2,N2,S2,L2
```

The parameters function in the following way:

K1,K2 - These are the data set reference numbers of the sets which will be changed by the trial program.

N1,N2 - These are the maximum record numbers for the trial subsets of the data sets. Usually they will be no greater than 40 or 50.

S1,S2 - These are the maximum record sizes of the two data set trial subsets. They must match the true record sizes of the actual data sets.

L1,L2 - These numbers are both optional. If L1 and L2 have the value 1, the corresponding subset will be listed (in hexadecimal format) as it appears after the program was run and as it appeared before the program was run. If the parameters have a value greater than 1, the first 10 records of the corresponding subset and 10 other records, starting with the specified record number, will be listed as they appeared after and before the program was run. If either parameter is missing, nothing is displayed for that subset.

The procedure uses file SCA as a temporary scratch file, so the user always avoids using it in the programs. The procedure also uses the name X for a temporary file name, so the use of this name is also avoided. The procedure follows. Note that the processing of the second parameter set mirrors that of the first.

```
$PROCEDURE TRIALR  
$EXE DAMP  
ASS RAD SCA X= NO  
MOVE X  
REFILE RAD,X  
ALLOCATE RAD,YY,%2,%3,U
```

```

DEFINE %1,%2,%3,U
COPY %1,YY
REMOVE %1
RENAME YY,%1
EXIT
$IFM %5,7
$EXE DAMP
ALLOCATE RAD,YY,%6,%7,U
DEFINE %5,%6,%7,U
COPY %5,YY
REMOVE %5
RENAME YY %5
EXIT
$EXECUTE MAIN,BI,LINK,NOMAP,2
$IFM %4,16
$IF %4=1,8
$EXE DAMP
DEFINE %1,%2,%3,U
DISPLAY %1,,10,H
DISPLAY %1,%4,10,H
ASSIGN RAD=PND
DISPLAY %1,,10,H
DISPLAY %1,%4,10,H
AVRECORD C1,5
$EXECUTE DAMP
DEFINE %1,%2,%3,U
DISPLAY %1,,,H
ASSIGN RAD=PND
DISPLAY %1,,,H
ASSIGN RAD=SCA
EXIT
$IFM %8,15
$IF %8=1,8

$EXE DAMP
DEFINE %5,%6,%7,U
DISPLAY %5,,10,H
DISPLAY %5,%8,10,H
ASSIGN RAD=PND
DISPLAY %5,,10,H
DISPLAY %5,%8,10,H
AVRECORD C1,5
$EXECUTE DAMP
DEFINE %5,%6,%7,U
DISPLAY %5,,,H
ASSIGN RAD=PND
DISPLAY %5,,,H
EXIT
$$

```

It would be a simple matter to modify this procedure to present the display of the files in the opposite order (original before modified) or to guarantee that RAD would finally be assigned to a particular file. It would be possible to create the temporary files on another partition on the user's disc if another dummy file name were selected.

B.2.2 Assigning Logical Unit Numbers To Data Sets With Arbitrary Names

An installation has established a collection of 22 named data sets semi-permanently resident on several system partitions. A direct access directory is recorded on the first sector of one of these partitions. A background default assignment of RAD to this partition was established when the system was generated. The system also features several other file names in the background file list, assigned at system generation time to each of the partitions containing the permanent data sets. These file names have the form SFc, where c is a character selected to distinguish between the files. Background users seldom if ever use these file names directly; they are used in the direct access directory and in the procedures below. When data sets were allocated on file SFA, the first four sectors of its partition were not used. The file name SF1 was assigned to the same partition to which the name RAD was assigned. The names of the data sets are the following:

PAY	PBI	TIM
DA	FEB	WHT
F70	F71	F72
TBT	OLD	JPG
GAN	RDL	GKB
BPH	EKG	MMP
TAX	SSN	MCS
BCM		

Most of these sets are packed together on the various SPc files. Two of them, TAX and DAT, are large enough to occupy a full partition. Rather than provide an SFc file name for these data sets, the data set names themselves serve as file names and were assigned at system generation time. The directory entries for these two data sets are as follows:

@DAT	DAT	0	1023
@TAX	TAX	0	1023

All of the other entries reference one of the SFc names; none of them use the name RAD.

Most of the background programs at the installation are written in FORTRAN. Two procedures were written to allow the FORTRAN programmer to assign his logical unit numbers to the permanent data sets. The first procedure is used to reset any existing assignments. This procedure is called IDSA for Initialize Data Set Assignments. The second procedure is called DSA and is used to make the assignments. It can have from one to four pairs of arguments and is referenced as follows:

\$DO DSA U1=S1, U2=S2, U3=S3, U4=S4

where the U's are the FORTRAN logical unit numbers and the S's are the names of the data sets. Another procedure is provided for those users who have their own directory which they want to use in conjunction with the system directory. This procedure uses the standard scratch file, SC. If the user needs the information on this file, he must reassign SC before using the procedure. The procedure is named AUD and has two parameters. The first is a file name (other than RAD) which the user has assigned to his directory. The second parameter is used to indicate whether the other two procedures have been used in this job. The procedure AUD can be used in place of IDSA in case the user has a directory similar to the system directory but containing different data set keys. However if AUD is used, it requires that the user's directory be no longer than two sectors, since it uses file SFA.

Procedure DSA uses another procedure, DSAN, which provides the bulk of the processing. Procedure DSA merely assures that RAD is assigned to SFA, the file used by IDSA and AUD. If a user wishes to use only his own directory, he can assign RAD to his directory and use DSAN directly.

The Job Control sequence given next was generated by a user on this system who had written a program using five direct access files. Data sets 8, 10, and 347 were to be assigned to the standard system data sets. Data set 650 was to be assigned to a scratch file and data set 910 was to refer to the set with that name in his own directory. The user's directory is on partition PNU.

```
$ASS X PNU, 650=SCA
$DO IDSA
$DO DSA 8=PBI, 10=SSN, 347=TAX
$DO AUD X,YES
```

The following sequence was to produce substantially the same result, except that the user's directory resembled the system directory in that all of its keys were CAN-coded values. The user wanted his file 910 to be the data set identified as @302 (not just decimal 302) in his directory.

```
$ASS X PNU 650 SCA
$DO AUD X,NO
$DO DSA 8=PBI,10=SSN,347=TAX,910=302
```

The three procedures are as follows:

```
$PROCEDURE IDSA
$EXECUTE DAMP
ASSIGN RAD=SFA
MOVE SF1
REFILE SFA,RAD
EXIT
$PROCEDURE AUD
$IF %2=YES,1
$DO IDSA
$EXECUTE DAMP
ASSIGN RAD=SC
MOVE %1
REFILE RAD,%1
ASSIGN RAD=SFA
APPEND SC
MOVE RAD,NO
EXIT
$PRODEFAULT DSA,,,,,,,,
$ASSIGN RAD=SFA
$DO DSAN %1,%2,%3,%4,%5,%6,%7,%8
$PROCEDURE DSAN
$EXECUTE DAMP
$IFP %7,P, RENAME @%8 %7
$IFP %5,P, RENAME @%6 %5
$IFP %3,P, RENAME @%4 %3
RENAME @%2 %1
EXIT
$$
```

NOTE: If more than four assignments are to be made, additional references can be made to DSA. Also note that DSA does not permit two different numbers to be assigned to a single named data set. If this ability is desired, each RENAME in procedure DSAN above can be changed to TWIN. The use of this directive causes the directory to grow. DAMP will ignore any directive which might cause a directory to overlay sets sharing the RAD file with it. The REFILE directive in procedure IDSA allows DAMP to detect directory overflows caused by too many TWIN directives.

Neither the original version of DSAN nor the modified version allows an operation similar to the file-to-file assignment operation. For example, the statement \$DO DSA 71=GAN, 400=OLD, 314=71 will not associate 314 with GAN. Such operations could be allowed if the modified version of DSAN were further modified by removing the commercial at signs (@). The at signs would then be required on the \$DO references to the procedure.

The statement above could be entered as follows:

```
$DO DSA 71=@GAN,400=@OLD,314=71
```

B.2.3 Cataloging Source Or Object Records

An installation has established at system generation time assignments for two files, DL1 and DL2. DL2 is assigned to a partition consisting of two cylinders of a removable disc medium. DL1 is assigned to a partition consisting of all the remaining cylinders on that device. These two file names are used to support the following set of procedures. The user can reassign either or both names if he wishes to set up smaller areas for use by the procedure.

Procedure INIT is referenced simply as follows:

```
$DO INIT
```

and is called whenever the contents of DL1 and DL2 are to be completely reinitialized. This procedure must have been invoked at least once for the current DL1 and DL2, before the procedures GET, REMOVE, or SAVE can be used.

Procedure SAVE is used to copy those physical records which comprise the first file on a rewindable device onto the DL files and to associate with the copy a three character name which can be used for retrieval. Copying is done using the DAMP directive COPY and so is based on the record sizes defined for the file's device, not on the actual size of the records of the file. The file mark after the file normally terminates the copy, although the other methods of terminating COPY are still effective. Since many files do not fall into the narrow range of types acceptable to SAVE, a procedure named CREATE is included. The CREATE procedure is referenced as follows:

```
$DO CREATE ANYFILE,REWFIL,TYPE
```

The parameters are interpreted as follows:

- ANYFILE - This is the name of the source or object file containing the data to be rewritten. This file normally is not rewound, allowing the user to position it as desired. The file can contain any data acceptable to the MAX IV Source Update or Library Update programs.
- REWFIL - This is the name of the scratch file onto which the data are to be

copied. This file should be suitable for use in SAVE. It will be rewound before copying begins. This file name can be used as parameter two in the SAVE procedure reference.

- TYPE - This parameter should be the letter S or the letter O to indicate whether ANYFILE contains source or object images. If TYPE is S, the Source Update program will be used to copy from ANYFILE to REWFILE. If TYPE is O, the Library Update program will be used.

There is a fourth parameter which indicates which of the two files is to be rewound before the copy is made. It should be omitted when CREATE is used to prepare files for SAVE. It is used by the GET procedure, which is described below.

The SAVE procedure is referenced as follows:

```
$DO SAVE KEYNAME,FILENAME
```

The two parameters are interpreted as follows:

- KEYNAME - This is a one- to three-character name which will be left justified (with trailing spaces), converted to CAN-code, and used to designate the particular data being saved.
- FILENAME - This is the name of the file from which the data to be saved are to be read.

Neither REWFILE nor FILENAME should be the name SCA, since the SAVE procedure uses SCA for temporary storage of data. All of these procedures also use the file names S1 and S3 internally.

Once any name has been established as a KEYNAME, it must not be used as a FILENAME in any of these procedures unless it is first deleted from the storage files (refer to the REMOVE procedure below).

Once data have been saved with the SAVE procedure, they can be recovered with the GET procedure. This procedure is referenced as follows:

```
$DO GET KEYNAME,IFILE,UFILE,TYPE
```

The parameters provide the following information:

- KEYNAME - This is a one- to three-character name specifying the particular data item which is to be retrieved. The name will be converted to CAN- code using standard system services.
- IFILE - This is the name of the file which is to receive the retrieved data. This file must have the same device characteristics as the file which contained the data when they were saved. The data will appear starting at the rewound position of this file and will be terminated by a file mark.

The next two parameters are optional as a set.

- UFILE - This is the name of a file which is to receive the data in a revised format. This file (and the TYPE) must be specified if the data should go

to a file whose characteristics are different from those of the file which contained the data when they were saved. UFILE can also be specified if the retrieved data should ultimately appear as a file following other files on a device. If UFILE is specified, the procedure first places the data on IFILE and then copies IFILE to UFILE.

TYPE - This parameter must appear if UFILE appears. It should be the letter S or the letter O to indicate that the data are source records or object records, respectively.

Procedure REMOVE is a rather simple procedure which deletes a named item from the storage files DL1 and DL2. It is activated as follows:

```
$DO REMOVE KEYNAME
```

KEYNAME is the one-to-three-character name of the item which is to be deleted.

NOTE: Most systems will not need to use CREATE or the recopying options of GET if disc files are being manipulated.

A listing of the four procedures follows:

```
$PROCEDURE CREATE
$IFM %3,17
$IFM %4,P,$REWIND %2
$IF %3=S,2
$IF %3=O,8
$AVRECORD CI,13
$MOVE SI=S1, SO=S3
$EXECUTE SED,,NOLO
ASSIGN SI=%1, SO=%2
COPY
EXIT
$MOVE S1=SI, S3=SO
$AVRECORD CI,6
$MOVE BI=S1, BO=S3
$EXECUTE LIB,,NOLO
ASSIGN BI=%1, BO=%2
COPY
EXIT
$MOVE S1=BI, S3=BO
$PROCEDURE SAVE
$MOVE SO=S1, LO=S3
$EXECUTE UPD,,NOLO
ASSIGN SO=SCA, LO=NO
REWIND SO, SCA
NOCOMPRESS
ADD 0
ALLOCATE RAD @%1 4 SPACES
ASSIGN AI=CI
COMPRESS RAD
ALLOCATE
WE OF LO
ASSIGN LO=NO, AI=CI
.EXIT
$MOVE S1=SO
```

```

$ASSIGN AI=CI
$AVRECORD AI,2
$EXECUTE DAMP,,AO
ASSIGN RAD=NO
REWIND SCA
AVRECORD SCA,6
REWIND %2
ASSIGN LO=SCA
COPY @%2 @RAD
DISPLAY @%2 1,1
REWIND %2
ASSIGN LO=NO, RAD=DL1
DEFINE 1,1,9,U
DEFINE 3,4,9,U
COPY 2,3
COPY 1,3
DEFINE 3,40,1,U
SET 3,13,21,,
SET 3,37,1,,
SET 3,38,1,,
SET 3,39
REMOVE @%1
REWIND SCA
ASSIGN S1=SCA
AVRECORD S1,3
AVRECORD CI,24
ASSIGN LO=SCA, AI=S1
REWIND S1
AVFILE S1
AVRECORD CI,4
ASSIGN LO=S3, AI=S1
ASSIGN LO=S3
COPY @%2 @%1
LIST
ASSIGN LO=NO, RAD=NO
REWIND SCA
SAVE DEFINE,SCA
ASSIGN RAD=DL2
REMOVE @%1
COMPRESS RAD
DEFINE @SCA,1,80
ASSIGN LO=S3
ALLOCATE RAD,@%1 1,80
ASSIGN LO=NO
COPY @SCA,@%1
AVRECORD CI,16
EXIT
$MOVE S3=LO
$PROCEDURE GET
$MOVE LO=S3
$EXECUTE DAMP
ASSIGN LO=NO, RAD=DL2
DEFINE @%1 1,80
REWIND SCA
COPY @%1 @SCA

```

```

REWIND %2
WEOF SCA
WEOF SCA
REWIND SCA
RESTORE SCA
ASSIGN RAD=DL1
COPY @%1 @%2
WEOF %2
EXIT
$MOVE S3=LO
$IFM %4,2
$REWIND %2
$DO CREATE %2,%3,%4,R
$PROCEDURE INIT
$MOVE LO=S3
$EXECUTE DAMP
ASSIGN LO=NO, RAD=DL2
CREATE 20
ASSIGN RAD=DL1
CREATE 23
ENTER 1,SCA,1,1
ENTER 3,SCA,3,3
EXIT
$MOVE S3=LO
$PROCEDURE REMOVE
$EXECUTE DAMP
ASSIGN RAD=DL1
REMOVE @%1
COMPRESS RAD
ASSIGN RAD=DL2
REMOVE @%1
COMPRESS RAD
EXIT
$$

```

B.3 GENERAL USES OF DAMP

For the following examples, assume that there exists a data set with key 95 containing 2000 records of 80 bytes each. A directory entry on RAD specifies that data set 95 resides on file whose sector size is 100 words.

1. To copy data set 95 to a magnetic tape on drive MT1 as part of a file saving procedure, the following directives could be used:

```

DEFINE 95,2000,80
ASSIGN SO=MT1
REWIND SO
COPY 95,SO

```

By criteria a., c., and e. in 4.2.2, data set 95 will be accessed from file XYZ, sectors 700-criteria b., d., and f., SO will be referenced on file SO (MT1). Its record size will be 128 words and 32767 records can be transmitted. When the COPY is completed, MT1 will contain two thousand 128-word records. The first 40 words of each record will be a record listed above, with the COPY changed to:

COPY 95,95

will restore the data from tape to the data set. The extraneous words on tape do not affect the operation.

2. Data set 95 will be copied to MT1 as in example 1, but the extraneous 88 words will not be recorded.

```
DEFINE 95,2000,80
DEFINE 95,2000,80
ASSIGN 95=MT1
REWIND 95
COPY 95,95
```

The parameters for 95 are determined as in example 1. The parameters for 95 are determined by criteria b., c., and f., so that two thousand 80-byte physical records can be transmitted to file 95.

3. Data set 95 will be copied to MT1 on a sector by sector basis without regard to logical records.

```
ASSIGN 95=MT1
REWIND 95
COPY 95,95
```

The parameters for 95 are determined as in example 1.

The parameters for 95 are determined by criteria a., d., and e., so that 100 word records will be read starting at sector 700 of file XYZ. No meaningful record number limit is established, so that according to point 2 under criterion e., the copy will stop with the message:

BOUNDARY ERROR

At this point, MT1 will hold 801 records of 128 words each. The first 100 words of each of the first 800 records are the desired data. Record 801 was produced after the boundary error and can be ignored. Since the number of sectors in 95 can be determined by examining the directory (with LIST), the boundary error can be avoided by adding either or both of the following directives:

```
DEFINE 95,800,100,U
DEFINE 95,800,100,U
```

As in example 2, if the second directive is used the extraneous 28 words per record are not written on 95.

4. It is desired to copy data set 95 to magnetic tape, blocked 20 logical records per physical record. The directives are analogous to those of example 2.

```
DEFINE 95,100,1600
DEFINE 95,100,1600
ASSIGN 95=MT1
REWIND 95
COPY 95,95
```

Suppose data set 95 contained only 1990 records so that its RAD directory entry were:

95 XYZ 700 1495

The directives above would still perform the desired function, although the BOUNDARY ERROR message would be generated. Record 100 would contain the last ten logical records followed by 400 irrelevant words.

NOTE: The tape produced by the above directives can be recopied to disc by reversing the arguments in the COPY directive, as discussed in example 1. The disc file can then be copied to tape deblocked as in example 2. The blocking factors of tapes cannot be changed in a single pass, tape-to-tape copy (because of criterion f.), but can be changed by this two step process. If the disc file is made sufficiently large and the input file has a file-mark record, the number of records parameter can be set arbitrarily large during the first pass and can be calculated from the output record count for the second pass.

B.4 CHAINED PARTITION RAD FILES

The following examples show very basic as well as complex uses of DAMP dealing with chained partition RAD files.

B.4.1 Simple Usages

```
$EXE DAMP
ASS RAD A
ENTER @$$$A SCA
ENTER @$$$K SCB
ENTER @$$$Z SCC
ENTER @$$$ LNK
```

produces a new partition structure as:

Partition	Dataset	File Name	Start Sector	End Sector
A	100	BO	0	999
	101	RAD	100	299
	@\$\$\$A	SCA	0	0
	@\$\$\$K	SCB	0	0
	@\$\$\$Z	SCC	0	0
	@\$\$\$	LNK	0	0
B	200	BI	0	999
	201	RAD	100	399
C	300	SI	0	999
	301	RAD	100	499
D	400	SO	0	999
	401	RAD	100	599

INDEX

\$NOP, 3-2

ACTION, 3-2
ALLOCATE, 2-1, 3-3
APPEND, 2-1, 3-5
ASCII, 2-2, 3-6
ASSIGN, 3-2
AVFILE, 3-2
AVRECORD, 3-2

BKFILE, 3-2
BKRECORD, 3-2

catalog
 source records, B-10
 object records, B-10

CHAINED , 2-2, 3-7
COPY, 2-2, 3-8
CREATE, 2-1, 3-9

data formats, 4-8
data set keys, 3-1, 4-1
data sets, 4-2
debug, programs, B-5
DEFINE, 2-2, 3-10
DEFINE table, 3-10, 3-28, 4-9
Directives
 Data Sets, 2-2
 Directory Manipulation, 2-1
 Job Control
 Operational Control, 2-2
 syntax
direct access directory, 4-8
directories
 combining, B-3
 saving, B-4
DISPLAY, 2-2, 3-11

ELIMINATE, 2-1, 3-13
ENTER, 2-1, 3-14
ERROR MESSAGES, A-1
EXAMPLES, B-1
EXIT, 2-2, 3-15

file, initializing a new, B-2
FILE USAGE, 4-1
FORTRAN Mechanism, 4-5
FUNCTIONAL DESCRIPTION, 1-1

GENERAL DESCRIPTION, 1-1

GENERAL INFORMATION, 3-1

INITIALIZE, 2-1, 3-16

keys, 4-1

LIMITATIONS, 4-10

LIST, 2-1, 3-17

MIMIC, 3-18

MOVE, 2-1, 3-19

NOVERIFICATION, 3-20

object records

 patching, B-2

 cataloging, B-10

ONE, 2-2, 3-21

PURGE, 2-1, 3-22

Rad Partition Usage, 4-3, 4-7

REFILE, 2-2, 3-23

REMOVE, 2-2, 3-24

RENAME, 2-2, 3-25

RERAD, 4-7

RESTORE, 2-2, 3-26

REWIND, 3-2

SAVE, 2-2, 3-28

SAVE/RESTORE ARCHIVAL FORMAT, 4-9

scratch files, simple direct access, B-1

SET, 2-2, 3-30

SYNTAX, 3-1

SYSTEM CONFIGURATIONS, 4-7

TWIN, 2-2, 3-31

TWO, 2-2, 3-32

WEOF, 3-2

The following assignments and execution:

```
ASS SCA B
ASS SCB NO
ASS SCC NO
ASS BI file required
ASS BO file required
ASS RAD A
(a) ASS 100 RAD
    ASS 101 RAD
    EXE FORT1
```

would cause the FORTRAN program to find data sets 100 and 101 immediately in partition A.

(b) If the last three lines had been:

```
ASS 200 RAD
ASS 201 RAD
EXE FORT2
```

the effect would be:

the data sets required (200 and 201) cannot be found in RAD (partition A) so the first of the declared chain entries is searched. This is @\$\$A which is associated with logical file SCA which is assigned to B. Therefore the data sets 200 and 201 are found in partition B. Data set 201 is stored on RAD, but this does not mean partition A; in the chained RAD context data set 201 is found on the current partition, i.e. B. The assignment of 201 is adjusted to reflect this.

(c) Dynamic Refile

```
$EXE DAMP
ASS RAD C
ENTER @$$A SCC
ENTER @$$Z LNK
```

produces a new partition structure as:

Partition	Data Set	File Name	Start Sector	End Sector
A	100	BO	0	999
	101	RAD	100	299
	@\$\$A	SCA	0	0
	@\$\$K	SCB	0	0
	@\$\$Z	SCC	0	0
	@\$\$	LNK	0	0
B	200	BI	0	999
	201	RAD	100	399
C	300	SI	0	999
	301	RAD	100	499
	@\$\$A	SCC	0	0
	@\$\$Z	LNK	0	0

D	400	SO	0	999
	401	RAD	100	599

The following assignments and execution:

```

ASS SCA B
ASS SCB C
ASS SCC D
ASS LNK D
ASS SI as required
ASS SO as required
ASS RAD A
ASS 300 RAD
ASS 301 RAD
EXE FORT3

```

would cause:

1. The RAD directory on partition A to be searched. The data sets 300 and 301 would not be found. As these examples have not yet defined that FORT3 has been recompiled and relinked with MAX IV D.2 revisions of the FORTRAN Run Time Library the standard default values for unfound data sets would be taken.
2. Assuming FORT3 has been recompiled and relinked with the relevant library: the data sets are not found on RAD so the first chained RAD partition is scanned; @\$\$A is pointing to B via SCA so this directory is searched without finding the datasets. The next chain entry defined @\$\$K is searched, and since this points to C both data sets are found.
NOTE: The search terminates on detecting the data set required so the redefinitions of @\$\$A and @\$\$Z are not actioned.
3. Assume that the data set assignments and execution were for a relinked version of FORT4: the search would take place as above until the partition associated with @\$\$K was searched (C). The directory search of C does not find the data sets but it does encounter a chain association of @\$\$A to SCC, which causes the @\$\$A entry in the list to be set to SCC. Similarly for @\$\$Z, i.e. @\$\$Z is not associated with logical file LNK.

At the end of the directory the data sets are still unfound therefore the next chain entry in the list is selected. Since the previous entry was "past" @\$\$A the only effect of this redefinition was to add a few microseconds to the search time. The next entry from the list is @\$\$Z which is associated with D via LNK.

B.4.2. More complex usage

The chain of partitions is reset for each data set search; this fact can be used to direct the search. In order to illustrate this a re-structure of the partitions used for examples is needed.

In addition the assignment of RAD can radically alter the course of the search.

```

$EXE DAMP
ASS RAD B
ENTER @$$B LNK
REMOVE 200

```

```

ENTER @$$$C SCC causes null entry caused by REMOVE
ENTER 200 BI 0 999
REFILE
EXIT

```

Produces a new partition structure as:

Partition		Data Set	File Name	Start Sector	End Sector
A	100	BO	0	999	
	101	RAD	100	299	
	@\$A		SCA	0	0
	@\$K		SCB	0	0
	@\$Z		SCC	0	0
	\$\$\$		LNK	0	0
B	\$\$\$C	SCC	0	0	
	201	RAD	100	399	
	@\$B		LNK	0	0
	200	BI	0	999	
C	300	SI	0	999	
	301	RAD	100	499	
	@\$A		SCA	0	0
	@\$Z		LNK	0	0
D	400	SO	10	999	
	401	RAD	100	599	

The following assignments and execution:

```

ASS SCA B
ASS SCB D
ASS SCC C
ASS LNK A
ASS BI }
ASS BO } as required
ASS SI }
ASS SO }
ASS RAD A
ASS 400 RAD
ASS 401 RAD
EXE FORT4

```

would cause searching of partitions A (RAD), B (@\$A, SCA), A (@\$B, LNK), C (@\$C, SCC) and D (@\$K, SCB). Note that it is possible to rescan the initial RAD directory. This would be of use only if the directory contains @\$ entries only and restoration of dynamic chains to a known state is needed. Note that any @\$ association to logical file name RAD is not placed in the list; the main reason is to avoid "unintentional" loops; DAMP excludes the specific generation of such an entry but it can occur as the result of a REFILE.

The result of ASS RAD B is left for the reader to evaluate.

Assuming that a new program FORT234 is available, which uses the same assignments as above, but data sets 201, 300 and 400, the 201 search would be A (RAD), B (@\$\$A, SCA) in which 201 is found.

The 300 search would be A (RAD), B (@\$\$A, SCA), A (@\$\$B, LNK), C (@\$\$C, SCC) and the 400 search would add D (@\$\$K, SCB) to this list.

Execution of the same program with ASS RAD B producest

for the 201 search B (RAD);

for the 300 search B (RAD), A (@\$\$B, LNK), C (@\$\$C, SCC) and the 400 search adds D (@\$\$K, SCB) to this sequence.

INDEX

\$NOP, 3-2

ACTION, 3-2
ALLOCATE, 2-1, 3-3
APPEND, 2-1, 3-5
ASCII, 2-2, 3-6
ASSIGN, 3-2
AVFILE, 3-2
AVRECORD, 3-2

BKFILE, 3-2
BKRECORD, 3-2

catalog
 source records, B-10
 object records, B-10

CHAINED, 2-2, 3-7
COPY, 2-2, 3-8
CREATE, 2-1, 3-9

data formats, 4-8
data set keys, 3-1, 4-1
data sets, 4-2
debug, programs, B-5
DEFINE, 2-2, 3-10
DEFINE table, 3-10, 3-28, 4-9

Directives
 Data Sets, 2-2
 Directory Manipulation, 2-1
 Job Control
 Operational Control, 2-2
 syntax

direct access directory, 4-8

directories
 combining, B-3
 saving, B-4

DISPLAY, 2-2, 3-11

ELIMINATE, 2-1, 3-13
ENTER, 2-1, 3-14
ERROR MESSAGES, A-1
EXAMPLES, B-1
EXIT, 2-2, 3-15

file, initializing a new, B-2
FILE USAGE, 4-1
FORTRAN Mechanism, 4-5
FUNCTIONAL DESCRIPTION, 1-1

GENERAL DESCRIPTION, 1-1

GENERAL INFORMATION, 3-1
INITIALIZE, 2-1, 3-16
keys, 4-1
LIMITATIONS, 4-10
LIST, 2-1, 3-17
MIMIC, 3-18
MOVE, 2-1, 3-19
NOVERIFICATION, 3-20
object records
 patching, B-2
 cataloging, B-10
ONE, 2-2, 3-21
PURGE, 2-1, 3-22
Rad Partition Usage, 4-3, 4-7
REFILE, 2-2, 3-23
REMOVE, 2-2, 3-24
RENAME, 2-2, 3-25
RERAD, 4-7
RESTORE, 2-2, 3-26
REWIND, 3-2
SAVE, 2-2, 3-28
SAVE/RESTORE ARCHIVAL FORMAT, 4-9
scratch files, simple direct access, B-1
SET, 2-2, 3-30
SYNTAX, 3-1
SYSTEM CONFIGURATIONS, 4-7
TWIN, 2-2, 3-31
TWO, 2-2, 3-32
WEOF, 3-2

Fold



NO POSTAGE
NECESSARY
IF MAILED
IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 3624 FT. LAUDERDALE, FL 33309

POSTAGE WILL BE PAID BY ADDRESSEE

MODULAR COMPUTER SYSTEMS
1650 W. McNAB ROAD
P.O. BOX 6099
FT. LAUDERDALE, FLORIDA 33310



Attention: TECHNICAL PUBLICATIONS, M.S. #85

Fold

 **MODCOMP**

Please comment on the publication's completeness, accuracy, and readability. We also appreciate any general suggestions you may have to improve this publication.

If you found errors in this publication, please specify the page number or include a copy of the page with your remarks.

Your comments will be promptly investigated and appropriate action will be taken. If you require a written answer, please check the box and include your address below.

9

Comments: _____

[illegible]

Manual Title _____

Manual Order Number _____ Issue Date _____

Name _____ Position _____

Company _____

Address _____

____ Telephone () _____



Corporate Headquarters
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel. (305) 974-1380, TWX: 310-372-7837

International Headquarters
MODULAR COMPUTER SERVICES, Inc., The Business Centre, Molly Millars Lane, Wokingham, Berkshire, RG11 2JQ, UK, Tel. 0734-786808, TLX: 851849149

Latin American Sales Headquarters
MODULAR COMPUTER SYSTEMS, Inc., 1650 West McNab Road, P.O. Box 6099, Ft. Lauderdale, FL 33310, Tel. (305) 975-6562, TLX: 3727852

Canadian Headquarters
MODCOMP Canada Ltd., 400 Matheson Blvd. East, Unit 24, Mississauga, Ontario, Canada L4Z 1N8, Tel. (416) 890-0666, TELEX: 05-961279

SALES & SERVICE LOCATIONS THROUGHOUT THE WORLD

The technical contents of this document, while accurate as of the date of publication, are subject to change without notice.

Printed in the U.S.A.